

Tutorial UART-WiFi Bridge

- 1) Clone: MatlabProcessing, Cpp_Application and UARTWiFiBridge folders on your laptop.
- 2) Follow the step on the README on the git to install Arduino with the espressif library.
- 3) Open Arduino to update the software:

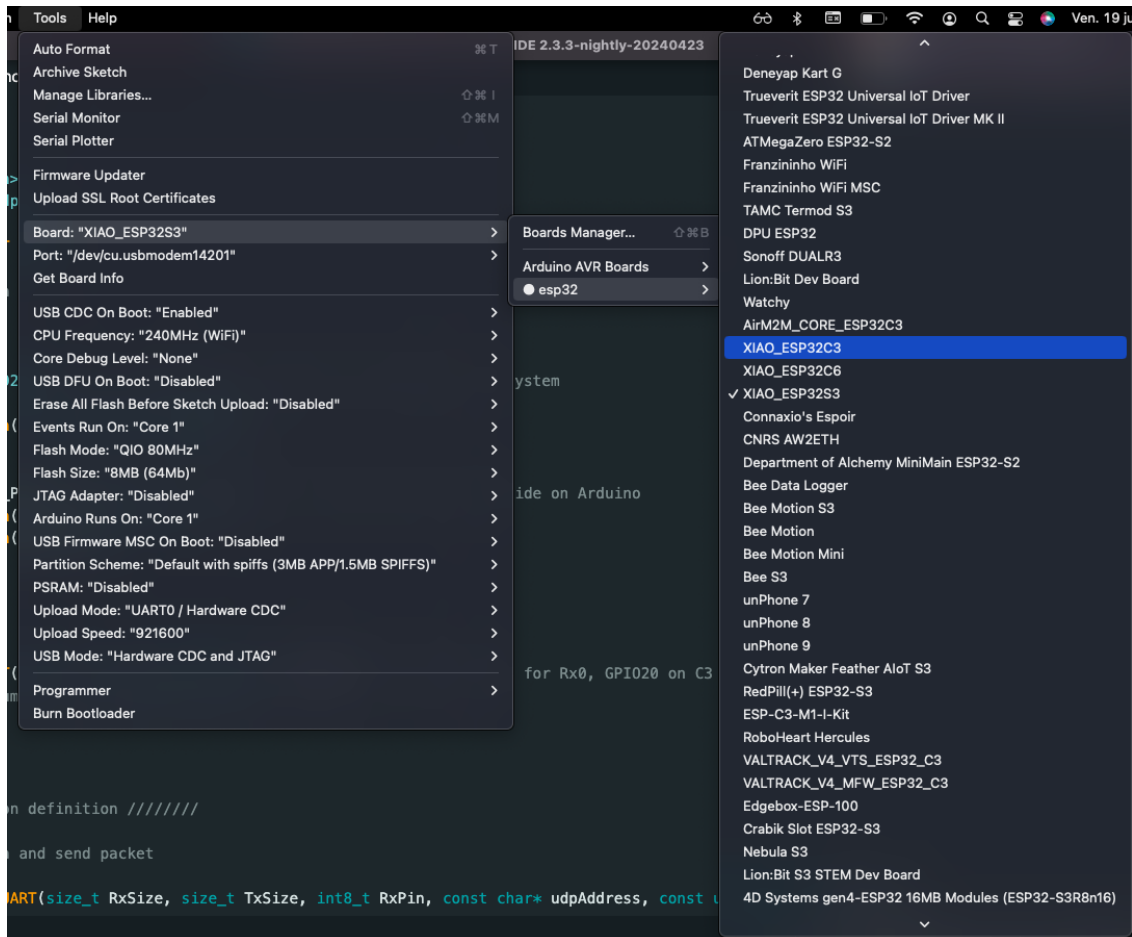


Figure 1: Board selection and port selection

If the esp32 tag is not present, the library is not well installed. When the board is selected, you can connect the ESP32 to your laptop using an USB and select the port.

If the board is detected by arduino, the board's name should be highlighted.

The UDPport should be define, the SSID and the password should correspond to the WiFi used and the IP precise in handle UART should be the IP of the receiver of the packets connected to the same WiFi of the ESP32.

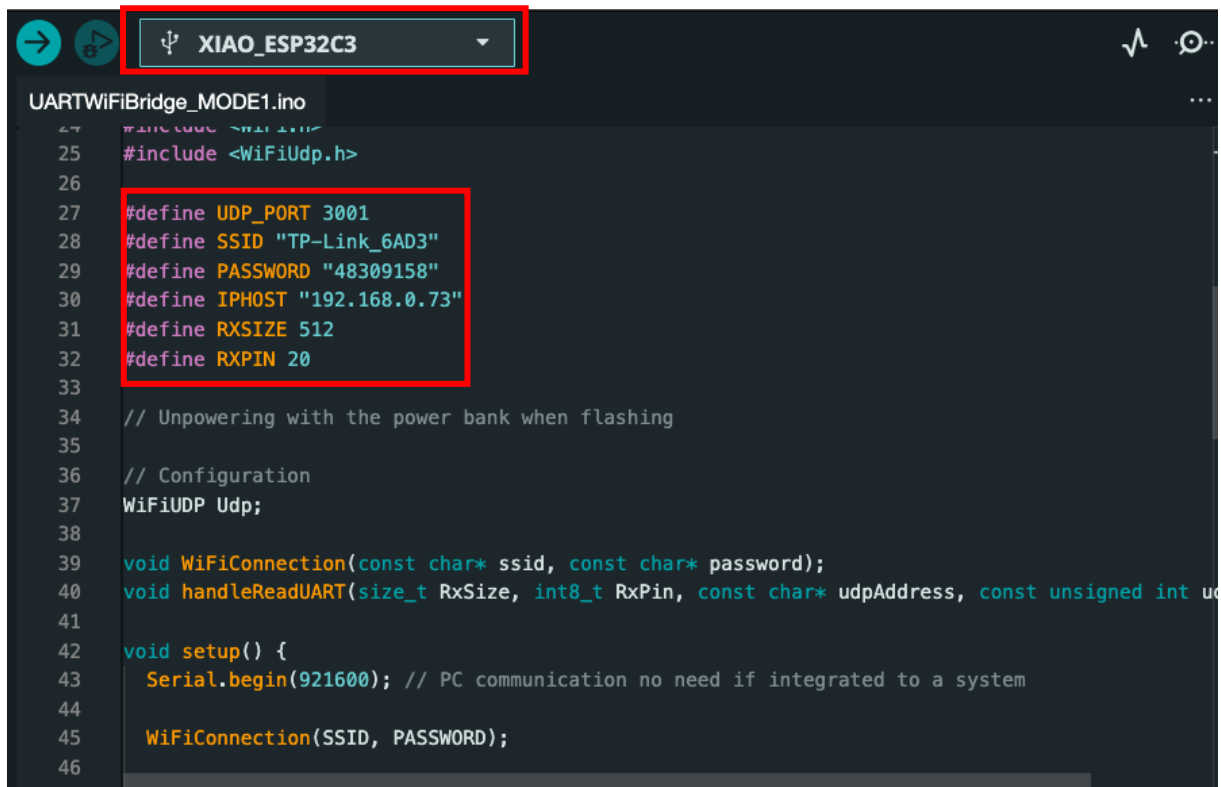


Figure 2: Arduino Window: parameters to define when flashing the ESP

UDP_PORT: send packets to this port

SSID: name of the WiFi

PASSWORD: WiFi password

IPHOST: IP of the device that receiving the packets

RXSIZE: size of the UART buffer size of the ESP32: 512 for C3 and 1024 for S3

RXPIN: pin of the receiver of the ESP32: 20 for C3 and 44 for S3

To get your IP: with a **Mac OSX** you can use **ifconfig** and with a **Windows** **ipconfig** just find the IPv4 when you are connected to the right WiFi.

```

[MacBook-Air-de-Theo:~ theobruneaudelasalle$ ifconfig
lo0: flags=8049<UP,LOOPBACK,RUNNING,MULTICAST> mtu 16384
    options=1203<RXCSUM,TXCSUM,TXSTATUS,SW_TIMESTAMP>
    inet 127.0.0.1 netmask 0xff000000
    inet6 ::1 prefixlen 128
    inet6 fe80::1%lo0 prefixlen 64 scopeid 0x1
    nd6 options=201<PERFORMNUD,DAD>
gif0: flags=8010<POINTOPOINT,MULTICAST> mtu 1280
stf0: flags=0<> mtu 1280
en0: flags=8863<UP,BROADCAST,SMART,RUNNING,SIMPLEX,MULTICAST> mtu 1500
    options=400<CHANNEL_IO>
    ether 98:46:0a:9b:9e:f6
    inet6 fe80::da:c737:efe1:5199%en0 prefixlen 64 secured scopeid 0x4
    inet 192.168.0.73 netmask 0xfffff00 broadcast 192.168.0.255
    nd6 options=201<PERFORMNUD,DAD>
    media: autoselect
    status: active
en1: flags=8963<UP,BROADCAST,SMART,RUNNING,PROMISC,SIMPLEX,MULTICAST> mtu 1500
    options=460<TS04,TS06,CHANNEL_IO>
    ether 82:18:67:7d:d4:c0
    media: autoselect <full-duplex>
    status: inactive

```

Figure 3: Find the IP for Mac OSX: ifconfig command in the terminal

- 4) Get the IP of the ESP in the Arduino terminal and check the ESP32 is connected to the WiFi

The screenshot shows the Arduino IDE interface. The top toolbar includes a dropdown menu set to 'XIAO_ESP32S3'. The main editor displays the code for 'UARTWifiBridge_Simulation.ino', with line 46 highlighted: `handleReadUART(1024, 0, 44, "192.168.0.73", UDP_PORT);`. The bottom panel shows the 'Serial Monitor' with the following output:

```

10:07:20.209 -> Connecting to WiFi...
10:07:20.209 -> Connected to WiFi:
10:07:20.209 -> IP address of the ESP32 (client):
10:07:20.209 -> 192.168.0.2
10:07:20.209 -> Server UDP bind to :
10:07:20.209 -> 4001

```

Figure 4: Arduino Serial after flashing the code into the chip

These two parameters will be used during the compilation of the C++ to use the application.

- 5) To check if the ESP32 is connected and receiving packets

Once you get the IP of the ESP32 and you flash the code, if you are using the same WiFi no need to flash again. You can use those two commands in the terminal to verify if the ESP32 is connected with the right IP and to the right WiFi. Make sure your computer and the ESP32 are connected to the same WiFi and use the command: **ping "IP"**

```
Air-de-Theo:~ theobrunaudelasalle$ ping 192.168.0.2
PING 192.168.0.2 (192.168.0.2): 56 data bytes
64 bytes from 192.168.0.2: icmp_seq=0 ttl=64 time=115.543 ms
64 bytes from 192.168.0.2: icmp_seq=1 ttl=64 time=31.436 ms
64 bytes from 192.168.0.2: icmp_seq=2 ttl=64 time=48.725 ms
64 bytes from 192.168.0.2: icmp_seq=3 ttl=64 time=67.981 ms
64 bytes from 192.168.0.2: icmp_seq=4 ttl=64 time=2.580 ms
^C
--- 192.168.0.2 ping statistics ---
5 packets transmitted, 5 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 2.580/53.253/115.543/37.845 ms
```

Figure 5: ping command to verify if the ESP32 is connected to the wifi: 0.0% packet loss means the ESP32 can exchange packets with the computer

If the ESP32 board UART is connected to the radar or IF module UART, data are sent to the UDP server. Before using Matlab or C++ you can check if the UDP port can be listened and the integrity of the data by using **netstat** command: **nc -ul UDPPORT**

```
[MacBook-Air-de-Theo:~ theobrunaudelasalle$ nc -ul 4001
FDA,FE8,FEB,FF4,000,000,000,000*53
$RT,082041,A,A,010323,932*1E
$RS,932,119,060,109,080*1A
$RH,932,01,FFA,FFB,FF9,000,003,008,000,FFC,FF8,000*59
$RH,932,02,004,007,FFF,FFB,FF8,FFF,002,006,002,FFF*29
$RH,932,03,FFA,FFC,000,006,005,FFF,FFA,FF8,FFD,003*58
$RH,932,04,008,000,FFC,FF9,FFF,004,007,FFE,FFA,FF8*2D
$RH,932,05,FFE,001,008,003,FFE,FF8,FFD,002,008,002*59
$RH,932,06,FFD,FFC,FFE,FFF,004,003,000,FFE,FFC,FFE*54
$RH,932,07,003,004,FFE,FFC,FFB,000,002,005,000,FFE*24
$RH,932,08,FFA,FFE,001,006,000,FFC,FF8,FFF,002,006*20
$RH,932,09,002,FFE,FF9,FFC,000,004,004,FFF,FFB,FFB*50
$RH,932,10,000,004,005,FFF,FFD,FFA,FFF,003,006,FFF*54
$RH,932,11,FFD,FFA,FFE,000,005,001,FFF,FFB,FFD,000*26
$RH,932,12,006,002,FFE,FF9,FFC,001,005,003,FFF,FFC*28
$RH,932,13,FFB,000,005,007,FFE,FFA,FF9,FFF,004,005*2A
```

Figure 7: UDP port 4001 listened: netstat command in the terminal

6) Fix the IP of the ESP32 and the (embedded) computer in WiFi settings

For the moment, the wifi router tp-link is used. To enter in the settings, connect your laptop to the TP-Link Wifi:

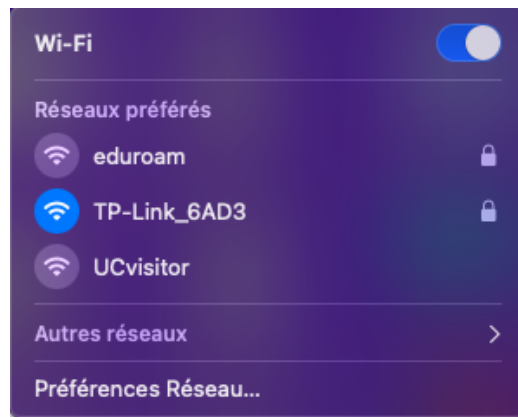


Figure 8: Connection to the Wifi: password can be found below the router

Wifi password: 48309158

Then open a navigator and enter: <http://tplinkwifi.net>

The LOCAL PASSWORD IS: WirelessTPLinkAX1800

Then going to: Advanced>Network>DHCP Server

tp-link | AX1800 Wi-Fi 6 Router

Search TP-Link ID Log Out

Network Map Internet Wireless **Advanced**

Quick Setup

Network

Status

Internet

LAN

IPTV/VLAN

• **DHCP Server**

Dynamic DNS

Routing

TP-Link ID

Address Reservation

Reserve IP addresses for specific devices connected to the router.

+ Add

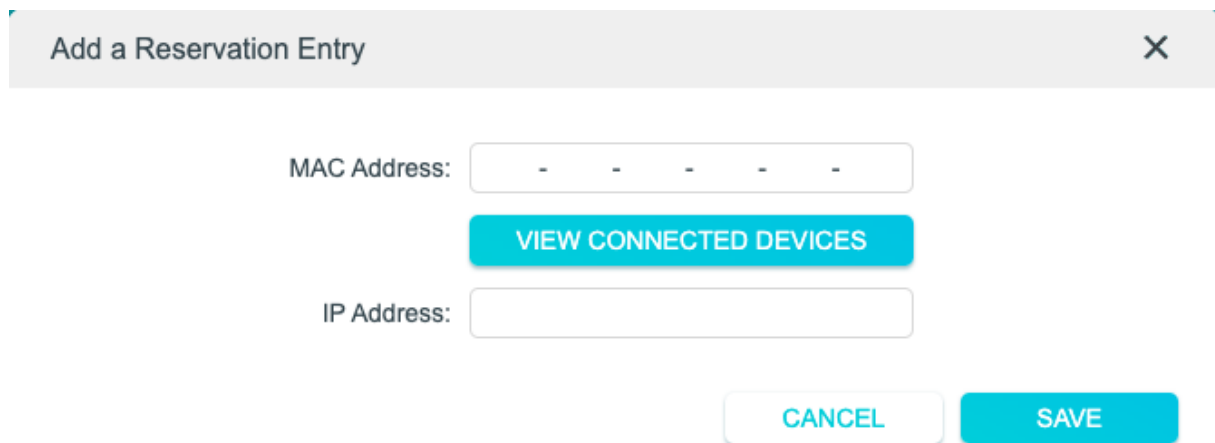
Device Name	MAC Address	Reserved IP Address	Status	Modify
esp32c3-047060	D4-F9-8D-04-70-60	192.168.0.3	<input checked="" type="checkbox"/>	Edit Delete
esp32s3-99558C	74-4D-BD-99-55-8C	192.168.0.2	<input checked="" type="checkbox"/>	Edit Delete

DHCP Client List

View the devices that are currently assigned with IP addresses by the DHCP server.

Figure 8: TPLink settings to fix the IP of the different devices

First connect the ESP32 to the WiFi, then click on “Add” in the DHCP Client List:



Add a Reservation Entry

MAC Address:

[VIEW CONNECTED DEVICES](#)

IP Address:

[CANCEL](#) [SAVE](#)

Figure 9: Reserved an IP for a device

Then click on “VIEW CONNECTED DEVICES”. If the ESP32 you want to fix the IP is connected to the WiFi, it should be listed. Select the device you want to fix the IP. Choose the IP between: 192.168.0.2-192.168.0.999.

7) Using the Matlab application:

- Open the MatlabProcessing folder
- Add to your path Basic_tools folder
- The Offline_package folder should be use if you have already a text file with data registered.

- To use the Online_processing folder: the first time you run the program: ACCEPT THE MESSAGE REGARDING SECURITY OF EXTERNAL CONNECTION to allow connection to udp server

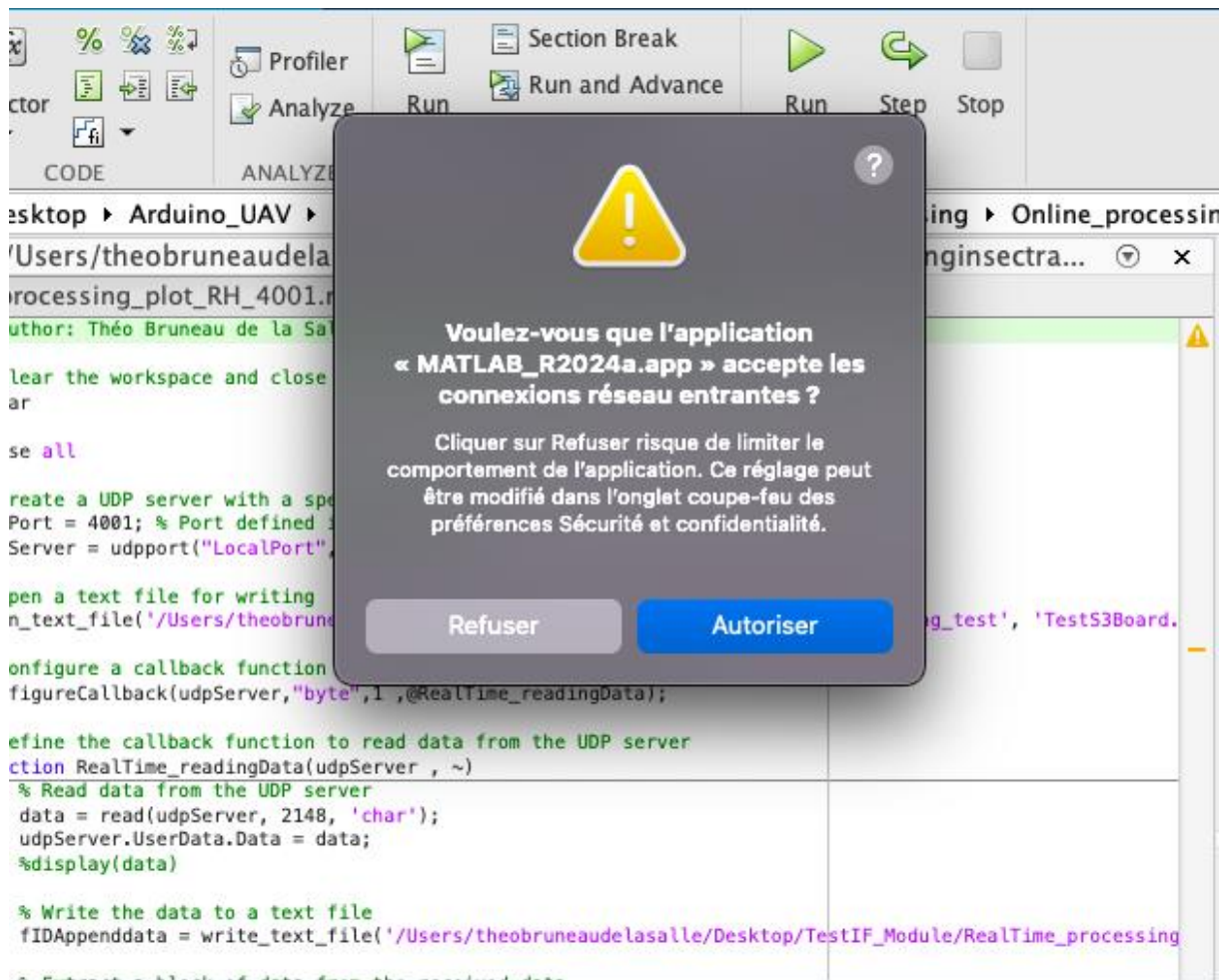


Figure 10: Warning message regarding the udp connection: click on authorize

Then the different files of the application can be used


```

8 % NbSweeps*2064* pourcentage: change the speed of the processing
9
10 clear
11 clc
12 close all
13
14 %UDP server creation
15 udpPort = 3001; %Port define in Arduino file
16 udpServer = udpport('LocalPort', udpPort);
17
18 configureCallback(udpServer, "byte", 1, @RealTime_readingDataV2);
19
20 function RealTime_readingDataV2(udpServer, ~)
21
22     persistent DataSec;
23     persistent typeofblock;
24     persistent NbSweeps;
25     typeofblock = 1;
26     NbSweeps = 33; %Number of blocks receive per second
27
28     data = read(udpServer, 2048, 'char');
29     udpServer.UserData.Data = data;
30     DataSec = [DataSec, data];
31
32     if (length(DataSec) > ceil(NbSweeps*2064*0.3)) %2064: Number of bytes in a full block
33         % Wait to have 30% of the data to process
34         [RadarDataBlock, RTGPS] = BlockExtraction(DataSec(ceil(NbSweeps*2064*0.2):end), '$RN'); %processing of the last 10% data recei
35         % RadarDataBlock n x 256 ready to be processed -> FFT
36
37         Parameters = RadarParameters(250e3, 1e-3, 3e8, 80e6, 0.78, 0.4);
38         WinHarmonicRadarDataBlockPad = TagRange_estimation(RadarDataBlock, Parameters, typeofblock, 5);
39         spectrum_plot(RadarDataBlock, Parameters, typeofblock, 0, 150); % 1 = Intermodule and 0 = HarmonicRadar
40         DataSec = [];
41     end
42 end
43

```

Figure 11: Real time visualization with Matlab

Those three parameters need to be changed depending on the application.

- If the UDP port define in Arduino is not 4001, change the **udpPort variable**.
- typeofblock = 1 for harmonic data and typeofblock = 0 for intermodule data
- '\$RH' when typeofblock = 1 and '\$RN' when typeofblock = 0

There are two files in the folder, one plotting the harmonic data and the other the intermodule data. Switch between those two files depending on the type of data to process and just change the **udpPort number**.

TO STORE THE DATA:


```

5   clc
6   close all
7
8   % Create a UDP server with a specific port
9   udpPort = 3001; % Port defined in Arduino file
10  udpServer = udpport("LocalPort", udpPort);
11
12  directory_rawdata = '/Users/theobrunaudelasalle/Desktop/TestIF_Module/';
13  filename_rawdata = 'TestS3Board.txt';
14
15  % Open a text file for writing
16  open_text_file(directory_rawdata, filename_rawdata);
17
18  % Configure a callback function to read data from the UDP server
19  configureCallback(udpServer, "byte", 1, @RealTime_readingData);
20
21  % Define the callback function to read data from the UDP server
22  function RealTime_readingData(udpServer, ~)
23
24      persistent directory_rawdata;
25      persistent filename_rawdata;
26      persistent DataSec; %Temporary buffer
27      persistent typeofblock;
28      persistent NbSweeps;
29
30      typeofblock = 0; %Type of block to process
31      NbSweeps = 33; %Number of blocks receive per second
32
33      directory_rawdata = '/Users/theobrunaudelasalle/Desktop/TestIF_Module/';
34      filename_rawdata = 'TestS3Board.txt';
35
36      % Read data from the UDP server
37      data = read(udpServer, 2148, 'char');
38      udpServer.UserData.Data = data;
39      DataSec = [DataSec, data];
40      % Write the data to a text file
41      fIDAppenddata = write_text_file(directory_rawdata, filename_rawdata, data);
42
43      if (length(DataSec) > ceil(NbSweeps*2064*0.3)) %2064: Number of bytes in a full block
44          % Wait to have 30% of the data to process
45          [RadarDataBlock RTCDPC] = BlockExtraction(DataSec(ceil(NbSweeps*2064*0.3):end), 'LRH'); %Processing of the last 10%

```

Figure 12: *Storing with Matlab*

TO SHUTDOWN THE APPLICATION: ENTER `configureCallback(udpServer, "off")` in Matlab terminal. Then click on the stop button once and if necessary twice.

8) C++ application usage

Go to the Cpp_application file on your laptop.

To create the executable, you need to follow those few steps:

- Using the cmake command:

This command has different options:

- `-Rx_THREAD = ON`: processing of Rx queue with $x = N, H, T, S$
- `-Rx_PLOT = ON`: plotting of Rx spectrum with $x = N, H$

For the plot: only one spectrum can be plotted and be sure that the corresponding thread is ON. By default, every option is OFF.

- `-DEXEC_NAME`: define the name of the executable
- `-DUDPPORT`: define the udp port to listen to: 4001
- `-DIPCLIENT`: define the IP of the client: ESP32 that is connecting to the radar: "192.168.0.4"
- `-DFULLPATH`: path to store the data: "/user/path/DataStorage.txt"

Below, an example of a compilation to listen to the **4001 port**, creation of a **CentralNode.txt** file and receiving the data from the ESP32: **"192.168.0.4"**, executable named **CentralNode**.

```
MacBook-Air:Cpp_application user$  
cmake -DEXEC_NAME=CentralNode -B build -DRN_THREAD=OFF -DRH_THREAD=ON -  
DRT_THREAD=OFF -DRS_THREAD=OFF -DRH_PLOT=OFF -DRN_PLOT=OFF -  
DFULLPATH="/Path/CentralNode.txt" -DUDPPORT=4001 -DIPCLIENT="192.168.0.94" .
```

Figure 12: creation of the makefile: compilation

-B build: Precise the construction folder. And **the point at the end is needed**.

After using this command, you need to create the executable by using make.

```
MacBook-Air:Cpp_Application user$ make -C build
```

Figure 13: creation of the executable

-C build: go to the directory to find the construction file

```
MacBook-Air:Cpp_Application user$ ./build/bin/CentralNode
```

Figure 14: launch the application: the executable is in bin directory