Mid-term Progress Report:

Submitted by: Nawaf Alzahem

|  |  |
| --- | --- |
| **Task:** | **Implementation Progress** |
| **Project Title:** | **Intelligent Autonomous Hoover as an IoT Solution** |
| **Team Members’ Names:** | 1. Nawaf Alzahem (Group Leader) 2. Fahad Aldulaigan 3. Mohammad Abuhaimed 4. Khalid Abu Alsaud 5. Khalid Hali |
| **Supervisor Name:** | **Dr.Niddal Nasser** |
| **Date:** | **12/3/2023** |

Index:

|  |  |
| --- | --- |
| **Project Management Plan** | **P.2 – P.3** |
| **Complete part 1 of Implementation development and Testing** | **P.4 – P.49** |
| **github link and bar code for our capstone project that stores all the continues progress achieved .** | **P.50 – P.51** |

1. Agile Methodology Project Management Plan

|  |  |
| --- | --- |
| **Requirement Collection producing SRS document** | **Done from last semester, but few modification needed in overlapping process due to implementation needs** |
| **Creating Software Design document** | **Done from last semester, but few modification needed in overlapping process due to implementation needs** |
| **Planning of Sprints** | **Heart of the work, arrangeable sprint continues iteration of development and testing between us to complete all phases of the project** |
| **Implementation development part 1 (WSN, Embedded Systems, and IoT In Waspmote IDE)** | **Completely Done** |
| **Implementation development part 2 (Android Application Development using Android Studio for the usability of biometrics)** | **Just started after finishing implementation of part1** |
| **Monitor and Review System** | **Haven’t reached it yet** |

A screenshot of a computer

Description automatically generated

1. part 1 of the implementation of IoT, Embedded Systems, and WSN codes completed In Waspmote IDE works well and passed each part of verification, compilation, uploading, and testing.

1: \_802\_15\_4\_TX\_FULL:

#include <WaspXBee802.h>

#include <WaspFrame.h>

/\* https://github.com/NawafAlzahem/Capstone-Project.git \*/

// TODO: Only broadcast seems to work

//char RX\_ADDRESS[] = "000000000000FFFF"; /\* BROADCAST \*/

char RX\_ADDRESS[] = "0013A20041678A0E"; /\* Robby1 MEshlium MAC \*/

//char RX\_ADDRESS[] = "0013A2004149DA14"; //XBEE S1 from IoT lab

//char RX\_ADDRESS[] = "0013A2004149DA23";

/\*

\* NOTE Nov 08, 2022:

\* Only NET address seems to work.

\* MAC Address works only for broadcast,

\* not for specific address

\*/

char MY\_NET\_ADDRESS[]="1220";

char RX\_NET\_ADDRESS[]="1221";

//char RX\_NET\_ADDRESS[]="1111"; // MESHLIUM NET ADDRESS

/\* In 802.15.4 you can use either full 8 HEX standard MAC

\* Address, or you can use the 2 HEX NET address.

\* The receiver will adapt automatically.

\* 0= use NET ADDRESS, 1 = USE MAC \*/

uint8\_t ADDRESS\_TYPE = 1;

char NODE\_ID[] = "TX\_SMART\_CITIES";

uint8\_t PANID[2] ; /\* Set the PAN\_ID global variable \*/

char node\_data[20];

uint8\_t error;

int counter = 0;

int delayTime;

uint8\_t api\_mode = 1; /\* 0-2 \*/

uint8\_t mac\_mode = 0; /\* 0-3 \*/

void setup(){

USB.ON();

USB.println(F("\n\*\*\*\*\*\*\* 802.15.4 SEND (TX) FULL \*\*\*\*\*\*\n"));

xbee802.ON();

PANID[0]=0x33;

PANID[1]=0x32;

/\* Only if you need to set the ID \*/

if(xbee802.setPAN(PANID)!=0){USB.println(F("ERROR setting PAN\_ID..."));}

error=xbee802.getPAN();

if(error!=0){ /\* Only if it didn't get the ID \*/

USB.println(F("ERROR getting PAN\_ID..."));

}

USB.print(F("---> PAN\_ID: "));

for(int i=0;i<2;i++){

USB.printHex(xbee802.PAN\_ID[i]);

}

USB.println();

xbee802.setChannel(0xC);

xbee802.getChannel();

USB.print(F("---> xbee802.channel: "));

USB.printHex(xbee802.channel); /\* IT SHOULD BE CHANNEL C \*/

USB.println();

xbee802.setNodeIdentifier(NODE\_ID);

xbee802.getNodeIdentifier();

USB.print(F("---> node ID: "));

USB.println(xbee802.nodeID); /\* Should print the whole array \*/

RTC.ON(); /\* Mote's time \*/

/\* TESTED, WORKS FINE! \*/

error = xbee802.setRTCfromMeshlium("0013A20041678A0E"); //Robby1

if( error != 0 ){ /\* Only in case of error...\*/

USB.println(F("SET RTC Time from Meshlium error..."));

}

USB.print(F("---> API mode: "));

USB.println(api\_mode, DEC);

/\* Set API mode accordingly \*/

xbee802.setAPI(api\_mode); /\* WHEN SET OTHERWISE, IT IS NOT WORKING !! \*/

/\* to select at any time if the modules are to use

\* a totally compatible heading format \*/

xbee802.setMacMode(mac\_mode); /\* 0-3 \*/

USB.print(F("---> MAC mode: "));

USB.println(xbee802.getMacMode(), DEC);

USB.print("---> My MAC ADDRESS: ");

xbee802.getOwnMacHigh();

xbee802.getOwnMacLow();

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacHigh[i]);

}

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacLow[i]);

}

USB.println();

USB.print(F("---> setOwnNetAddress("));

USB.print(MY\_NET\_ADDRESS);

if(xbee802.setOwnNetAddress(MY\_NET\_ADDRESS) == 0){

USB.print(F(") SUCCESSFUL..."));

} else{

USB.print(F(") ERROR!..."));

}

USB.println();

// Show '\_serial\_id' stored by the API when powering up

USB.print(F("---> Serial ID: "));

for (int i=0;i<8;i++){

USB.printHex(\_serial\_id[i]);

}

USB.println();

USB.print(F("---> RTC Time:"));

USB.println(RTC.getTime());

}

void loop(){

USB.println(F("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* LOOP STARTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

counter++;

sprintf(node\_data,"Packet No:%d",counter);

USB.println(F("-------- 1. Create ASCII frame ----------"));

frame.createFrame(ASCII, NODE\_ID); /\* Set the Node ID for the frame \*/

frame.setFrameSize(125);

/\* EXAMPLE\_FRAME

TIMEOUT\_FRAME

EVENT\_FRAME

ALARM\_FRAME

SERVICE1\_FRAME

SERVICE2\_FRAME \*/

frame.setFrameType(EVENT\_FRAME);

frame.addSensor(SENSOR\_STR, node\_data);

frame.addSensor(SENSOR\_BAT, PWR.getBatteryLevel());

//USB.println(F("---> Frame 2 send: ")); frame.showFrame();

if(ADDRESS\_TYPE == 0){ /\* 0 or 1 \*/

USB.print(F("Sending to NET address: "));

USB.println(RX\_NET\_ADDRESS);

error = xbee802.send(RX\_NET\_ADDRESS, frame.buffer, frame.length);

}else{

USB.print(F("Sending to MAC address: "));

USB.println(RX\_ADDRESS);

error = xbee802.send(RX\_ADDRESS, frame.buffer, frame.length);

}

if( error == 0 ){ // check TX flag

USB.println(F("---> frame sent ok"));

Utils.blinkGreenLED();

/\* it should deteriorate with distance \*/

USB.print(F("---> RSSI(dBm): "));

USB.println(xbee802.\_rssi);

/\* Receive XBee packet (wait message for 2 seconds) \*/

error = xbee802.receivePacketTimeout( 4000 );

if( error == 0 ){

USB.println(F("------- 1.1 Responce packet received ---------"));

/\* define the variable ad hoc \*/

char data\_in[xbee802.\_length]; // assign the length correct value

/\* Show data stored in '\_payload' buffer indicated by '\_length' \*/

USB.print(F("---> Received message: "));

for(int i=0;i<xbee802.\_length;i++){

USB.print(xbee802.\_payload[i]);

}

USB.println();

}

} else {

USB.println(F("---> OOPS..., frame sent error"));

Utils.blinkRedLED();

}

delayTime = 5; /\* seconds \*/

USB.print(F("\n\*\*\*\*\* LOOP END, Wait for "));

USB.print(delayTime);

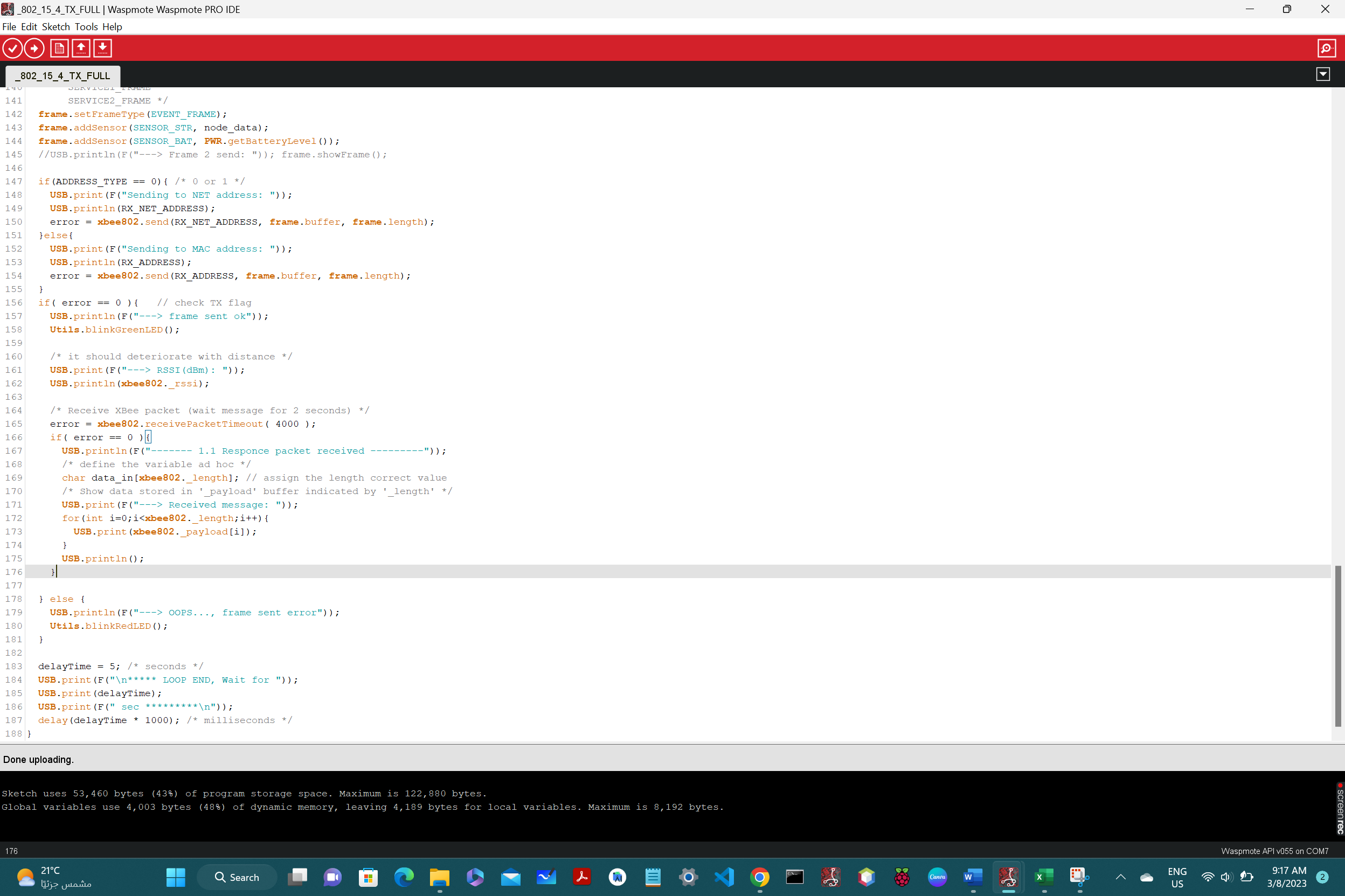
USB.print(F(" sec \*\*\*\*\*\*\*\*\*\n"));

delay(delayTime \* 1000); /\* milliseconds \*/

}

Detailed explanation of the functionality of the code:

The code is for sending data over a wireless network using the 802.15.4 protocol through a Waspmote device. The code initializes some variables and defines the networking parameters such as RX\_ADDRESS, MY\_NET\_ADDRESS, RX\_NET\_ADDRESS, and ADDRESS\_TYPE. Then in the setup() function, it first turns on the Waspmote device and sets the PANID (the id of the network the device is connected to). Next, it sets up the node identifier, the RTC time of the device and the API mode of the device. After that, it checks details such as the Mac mode, own MAC address, Own net address and Serial ID of the device. In the loop() function, it creates an ASCII frame, sends it to the specified address (either with the MAC address or NET address) and then receives the response packet before waiting for some time before starting the loop again.



2: \_802\_11b\_RX\_FULL:

#include <WaspXBee802.h>

#include <WaspFrame.h>

uint8\_t error;

uint8\_t TX\_MAC\_ADDRESS[8]; // CAN BE NOT USED in 802.15.4 ?

uint8\_t api\_mode = 1; /\* 0-2 \*/ // ONLY 1,2 seem to work!

uint8\_t mac\_mode = 0; /\* 0-3 \*/ // when API=1 or 2, mac is 0 no matter what

int delayTime;

char WASPMOTE\_ID[] = "4242\_RECV\_RX";

char MY\_NET\_ADDRESS[] = "4242";

char TX\_NET\_ADDRESS[2];

int rssi;

void setup(){

USB.ON();

USB.println(F("\*\*\*\*\* Complete example (RX node) \*\*\*\*\*\n"));

xbee802.ON();

RTC.ON(); /\* Mote's time \*/

/\* TESTED, WORKS FINE! \*/

error = xbee802.setRTCfromMeshlium("0013A20041678A0E"); //Robby1 MAC

if( error == 0 ){

USB.println(F("SET RTC Time from Meshlium is ok! \n"));

}else{

USB.println(F("SET RTC Time from Meshlium error..."));

}

error=xbee802.getPAN();

if(error!=0){ /\* Only if it didn't get the ID \*/

USB.println(F("ERROR getting PAN\_ID..."));

}

USB.print(F("---> PAN\_ID: "));

for(int i=0;i<2;i++){

USB.printHex(xbee802.PAN\_ID[i]);

}

USB.println();

xbee802.setNodeIdentifier(WASPMOTE\_ID);

xbee802.getNodeIdentifier();

USB.print(F("---> node ID: "));

USB.println(xbee802.nodeID); /\* Should print the whole array \*/

USB.print(F("---> API mode: "));

USB.println(api\_mode, DEC);

xbee802.setAPI(api\_mode); /\* WHEN SET OTHERWISE, IT IS NOT WORKING !! \*/

/\*to select at any time if the modules are to use

\* a totally compatible heading format \*/

xbee802.setMacMode(mac\_mode);

USB.print(F("---> Active MAC mode: "));

USB.println(xbee802.getMacMode(), DEC);

USB.print("---> My MAC ADDRESS: ");

xbee802.getOwnMacHigh();

xbee802.getOwnMacLow();

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacHigh[i]);

}

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacLow[i]);

}

USB.println();

xbee802.getChannel();

USB.print(F("---> xbee802.channel: "));

USB.printHex(xbee802.channel); // currently C

USB.println();

USB.print(F("---> setOwnNetAddress("));

USB.print(MY\_NET\_ADDRESS);

if(xbee802.setOwnNetAddress(MY\_NET\_ADDRESS) == 0){

USB.print(F(") SUCCESFULL..."));

} else{

USB.print(F(") ERROR!..."));

}

USB.println();

// Show '\_serial\_id' stored by the API when powering up

USB.print(F("---> Serial ID: "));

for (int i=0;i<8;i++){

USB.printHex(\_serial\_id[i]);

}

USB.println();

USB.print(F("---> RTC Time:"));

USB.println(RTC.getTime());

}

void loop(){

USB.println(F("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* LOOP STARTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

RTC.ON(); /\* Mote's time \*/

/\* Receive XBee packet (wait message for 2 seconds) \*/

error = xbee802.receivePacketTimeout( 4000 );

if( error == 0 ){

USB.println(F("------- 1. New packet received ---------"));

// TODO: Change the string in the sent packet and see if it changes

USB.print(F("---> Received packet length: "));

USB.println(xbee802.\_length); // uint16\_t \_length;

/\* define the variable ad hoc \*/

char data\_in[xbee802.\_length]; // assign the length correct value

/\* Show data stored in '\_payload' buffer indicated by '\_length' \*/

USB.print(F("---> Received message: "));

for(int i=0;i<xbee802.\_length;i++){

USB.print( xbee802.\_payload[i]);

data\_in[i] = xbee802.\_payload[i]; // use the data further

}

USB.println();

USB.print(F("---> Source (TX) MAC address: "));

for (int i=0;i<8;i++){

TX\_MAC\_ADDRESS[i] = xbee802.\_srcMAC[i];

USB.print(TX\_MAC\_ADDRESS[i], HEX);

}

USB.println();

USB.print(F("---> Source (TX) Network address: "));

TX\_NET\_ADDRESS[0]=xbee802.\_srcNA[0];

USB.print(xbee802.\_srcNA[0], HEX);

TX\_NET\_ADDRESS[1]=xbee802.\_srcNA[1];

USB.print(xbee802.\_srcNA[1], HEX);

USB.println();

USB.print(F("---> RSSI(dBm): "));

USB.println(xbee802.\_rssi);

/\* RSSI again \*/

xbee802.getRSSI();

if(!xbee802.error\_AT){

USB.print(F("---> getRSSI(dBm): "));

rssi=xbee802.valueRSSI[0];

rssi\*=-1;

USB.println(rssi,DEC);

}

USB.println();

/\* Wait TX node to prepare to receive messages \*/

//delay(1000);

/\* Send message back to TX node \*/

USB.println(F("\n------- 2. Send a response to the TX node ------"));

frame.createFrame(ASCII, WASPMOTE\_ID); /\* Set the ID inside the frame \*/

frame.setFrameSize(125);

/\* EXAMPLE\_FRAME

TIMEOUT\_FRAME

EVENT\_FRAME

ALARM\_FRAME

SERVICE1\_FRAME

SERVICE2\_FRAME \*/

frame.setFrameType(EVENT\_FRAME);

frame.addSensor(SENSOR\_BAT, PWR.getBatteryLevel());

USB.println(F("--->Frame to be sent back below"));

frame.showFrame();

// TODO: THE PACKET IS NOT SEND SUCCESFULLY in both cases !!!

if(TX\_NET\_ADDRESS){ // if empty address not used

error=xbee802.send(TX\_NET\_ADDRESS, frame.buffer, frame.length );

USB.print(F("Sent to NET Address: "));

USB.print(TX\_NET\_ADDRESS[0], HEX);

USB.print(TX\_NET\_ADDRESS[1], HEX);

if( error == 0 ){ /\* check TX flag \*/

USB.println(F(" SUCCESSFUL..."));

Utils.blinkGreenLED();

}else{

USB.println(F(" ERROR..."));

Utils.blinkRedLED();

}

}

if(TX\_MAC\_ADDRESS){ // if empty address not used

error=xbee802.send(TX\_MAC\_ADDRESS, frame.buffer, frame.length );

USB.print(F("Sent to MAC Address: "));

for(int i=0; i<8; i++){

USB.print(TX\_MAC\_ADDRESS[i], HEX);

}

if( error == 0 ){ /\* check TX flag \*/

USB.println(F(" SUCCESSFUL..."));

Utils.blinkGreenLED();

}else{

USB.println(F(" ERROR..."));

Utils.blinkRedLED();

}

}

}else{

/\* Print error message:

\* '7' : Buffer full. Not enough memory space

\* '6' : Error escaping character within payload bytes

\* '5' : Error escaping character in checksum byte

\* '4' : Checksum is not correct

\* '3' : Checksum byte is not available

\* '2' : Frame Type is not valid

\* '1' : Timeout when receiving answer

\*/

USB.print(F("\tError receiving a packet:"));

USB.println(error,DEC);

}

delayTime = 5; /\* seconds \*/

USB.print(F("\n\*\*\*\*\* LOOP END, Wait for "));

USB.print(delayTime);

USB.print(F(" sec \*\*\*\*\*\*\*\*\*\n"));

delay(delayTime \* 1000); /\* milliseconds \*/

}

Detailed explanation of the functionality of the code:

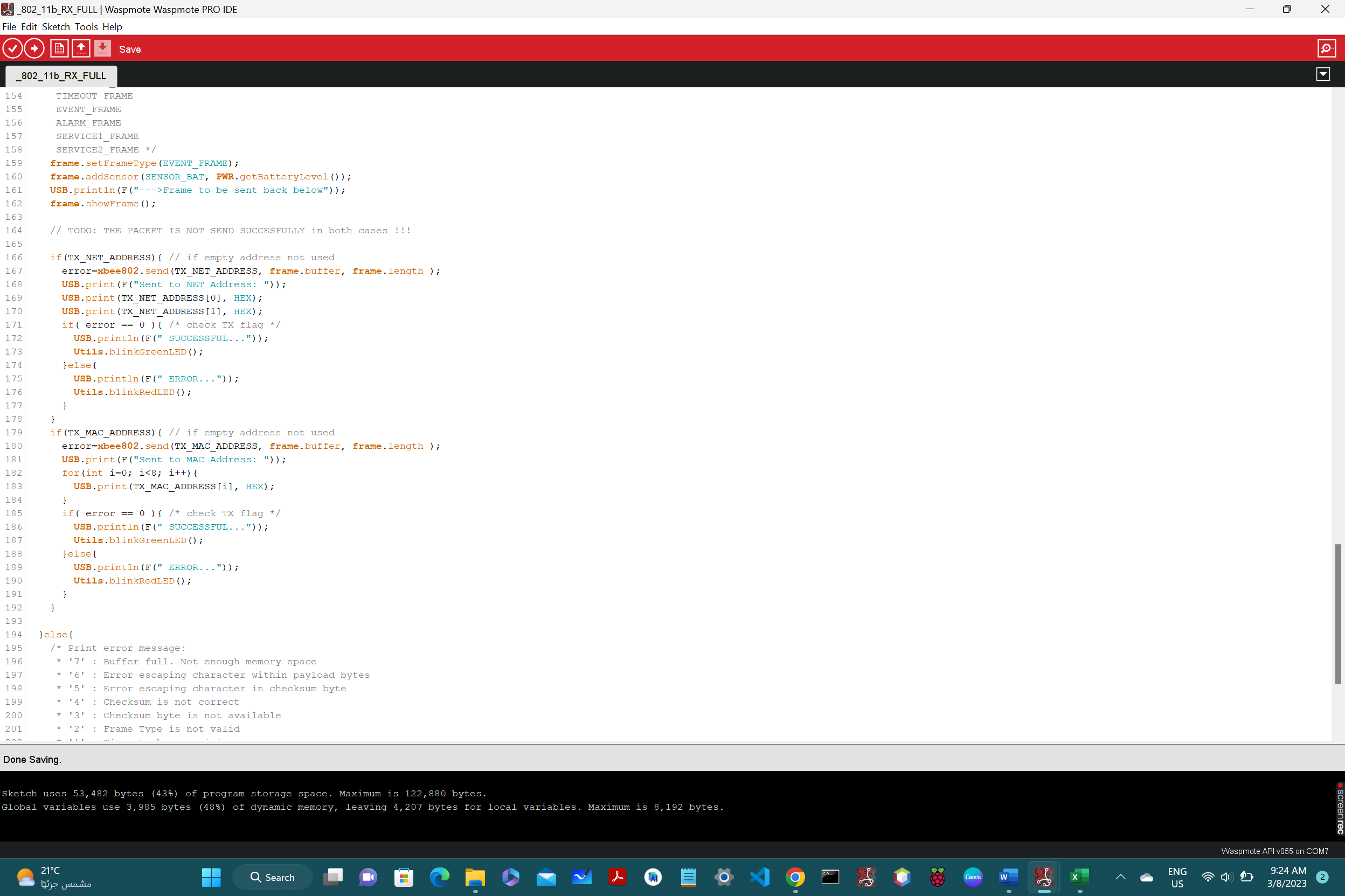
This code for the receive node of a wireless sensor module. it uses the WaspXBee802 and WaspFrame libraries.

The main purpose of the code is to receive data packets from a transmitting node and send a response back to the transmitter if required. The code first declares various global variables required for the program, such as error flags and network addresses.

In the setup() function, the code initializes the USB and XBee802 modules and sets the time of the device. It then retrieves the PAN ID of the network and sets the node identifier. It also sets the API and MAC modes of the device and retrieves its own MAC address and channel. Finally, it sets its own network address and prints the serial ID and RTC time.

In the loop() function, the code waits for a packet to arrive from the transmitter by calling the receivePacketTimeout() function. If a packet is received successfully, the code retrieves various details about the packet, such as its length, message data, source MAC and network addresses, and RSSI value. It then sends a response back to the transmitter using the send() function and prints the respective success/failure message.

If no packet is received within the timeout period, the code prints an error message. The loop then waits for a specified time period before repeating the process.



3: WIFI\_PRO\_16b\_send\_to\_meshlium\_via\_https:

#include <WaspWIFI\_PRO.h>

#include <WaspFrame.h>

// choose socket (SELECT USER'S SOCKET)

uint8\_t socket = SOCKET0;

char type[] = "https";

char host[] = "10.10.10.1";

char port[] = "443";

char TRUSTED\_CA[] =

"-----BEGIN CERTIFICATE-----\r"\

"MIIDxTCCAq2gAwIBAgIJANmfNvvZHqt5MA0GCSqGSIb3DQEBCwUAMHgxCzAJBgNV\r"\

"BAYTAkVTMQ8wDQYDVQQIDAZBcmFnb24xETAPBgNVBAcMCFphcmFnb3phMTIwMAYD\r"\

"VQQKDClMaWJlbGl1bSBDb211bmljYWNpb25lcyBEaXN0cmlidWlkYXMgUy5MLjER\r"\

"MA8GA1UEAwwIbWVzaGxpdW0wIBcNMTgwODIzMDcxNDE0WhgPMjExODA3MzAwNzE0\r"\

"MTRaMHgxCzAJBgNVBAYTAkVTMQ8wDQYDVQQIDAZBcmFnb24xETAPBgNVBAcMCFph\r"\

"cmFnb3phMTIwMAYDVQQKDClMaWJlbGl1bSBDb211bmljYWNpb25lcyBEaXN0cmli\r"\

"dWlkYXMgUy5MLjERMA8GA1UEAwwIbWVzaGxpdW0wggEiMA0GCSqGSIb3DQEBAQUA\r"\

"A4IBDwAwggEKAoIBAQDpWNu/u/UelrO6KsR0tofdhplr3npzm4ZqxvpvqNmYGykP\r"\

"NaelXnr/1Sl/39EYfcuwsUX4xsQSBcHrtm6HVrq1+vtYC/v8CV02nKCc0NtwR3nk\r"\

"KuF41Yy56kXKOE0LRkhcrI0Ti67apApN/v7NDuXU7j404eTiRACMGiL+HkUkO2Wr\r"\

"/YtMjsdDwgxDRet9GeCv/Euqy2xirjv1nuU5rG6uoJzWauXCvekutMMJ/2XXtweF\r"\

"X7SzWIkRz3LcOBXRhafs/dO1aL9LVibtDOxUYyLvJyeyL6FKjpqRHKAvehfsR49N\r"\

"VkLkBCl/tjhcTTHpvm0bsG+q7aKLoOhp8lYItDfLAgMBAAGjUDBOMB0GA1UdDgQW\r"\

"BBQpsed68QeqW+nIaXun/FOvVm4WzjAfBgNVHSMEGDAWgBQpsed68QeqW+nIaXun\r"\

"/FOvVm4WzjAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQB3rXMiBUx7\r"\

"yeJO29xKSAgdrGZe189WWiCfvdphMuaSH8OQooKd5d4a6CzmoiklusAFTfGiaXqK\r"\

"DPuE5/atjLAluj4NR2MZ4ti9bqhcCFdlDGKXAj3wqvEIGbjVppXkPRlFyM+/VIW6\r"\

"7hm59xKsWyymqxBfKmFsXgFA90mOqmzoPV0pu2d5Q+Kh6gEZYauiWkLIxa+y36Wq\r"\

"+HCe1q3vch5zOhPUnlpUmhzQd4K/AlyHEZUnG0hPzURnXk+qspb5xsiHuRoj9mkh\r"\

"MTNKb37FmwyZyM9u+KuNRKvZ0TJY0zs9+SDoDQJVRMwGh7I6IEHCuWNMl4N0aNaY\r"\

"2huxbcU1t/Z+\r"\

"-----END CERTIFICATE-----";

uint8\_t error;

uint8\_t status;

unsigned long previous;

char moteID[] = "nawaf\_https";

void setup(){

USB.println(F("Start program"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.println(F("Once the module is set with one or more"));

USB.println(F("AP settings, it attempts to join the AP"));

USB.println(F("automatically once it is powered on"));

USB.println(F("Refer to example 'WIFI\_PRO\_01' to configure"));

USB.println(F("the WiFi module with proper settings"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

frame.setID(moteID);

error = WIFI\_PRO.ON(socket);

if (error == 0){

USB.println(F("1. WiFi switched ON"));

}else{

USB.println(F("1. WiFi did not initialize correctly"));

}

error = WIFI\_PRO.setCA(TRUSTED\_CA);

if (error == 0){

USB.println(F("2. Trusted CA set OK"));

}else{

USB.println(F("2. Error calling 'setCA' function"));

WIFI\_PRO.printErrorCode();

}

//WIFI\_PRO.OFF(socket);

//USB.println(F("3. WiFi switched OFF"));

//USB.println(F("Setup done\n\n"));

}

void loop()

{

// get actual time

previous = millis();

error = WIFI\_PRO.ON(socket);

if (error == 0){

USB.println(F("WiFi switched ON"));

}else{

USB.println(F("WiFi did not initialize correctly"));

}

// 2. Join AP

status = WIFI\_PRO.isConnected();

if (status == true){

USB.print(F("WiFi is connected OK"));

USB.print(F(" Time(ms):"));

USB.println(millis() - previous);

// create new frame (only ASCII)

frame.createFrame(ASCII);

frame.addSensor(SENSOR\_STR, "this\_is\_a\_string");

frame.addSensor(SENSOR\_BAT, PWR.getBatteryLevel());

frame.showFrame();

// 3.2. Send Frame to Meshlium

// http frame

error = WIFI\_PRO.sendFrameToMeshlium( type, host, port, frame.buffer, frame.length);

if (error == 0){

USB.println(F("HTTP OK"));

USB.print(F("HTTP Time from OFF state (ms):"));

USB.println(millis() - previous);

}else{

USB.println(F("Error calling 'getURL' function"));

WIFI\_PRO.printErrorCode();

}

}else{

USB.print(F("WiFi is connected ERROR"));

USB.print(F(" Time(ms):"));

USB.println(millis() - previous);

}

// 3. Switch OFF

//WIFI\_PRO.OFF(socket);

//USB.println(F("WiFi switched OFF\n\n"));

delay(3000);

}

Detailed explanation of the functionality of the code:

This code is designed to connect a Waspmote board to a Wi-Fi network and send data to a server using HTTPS protocol. Here is a brief explanation of each part of the code:

- First, the necessary libraries are included for Wi-Fi and frame handling.

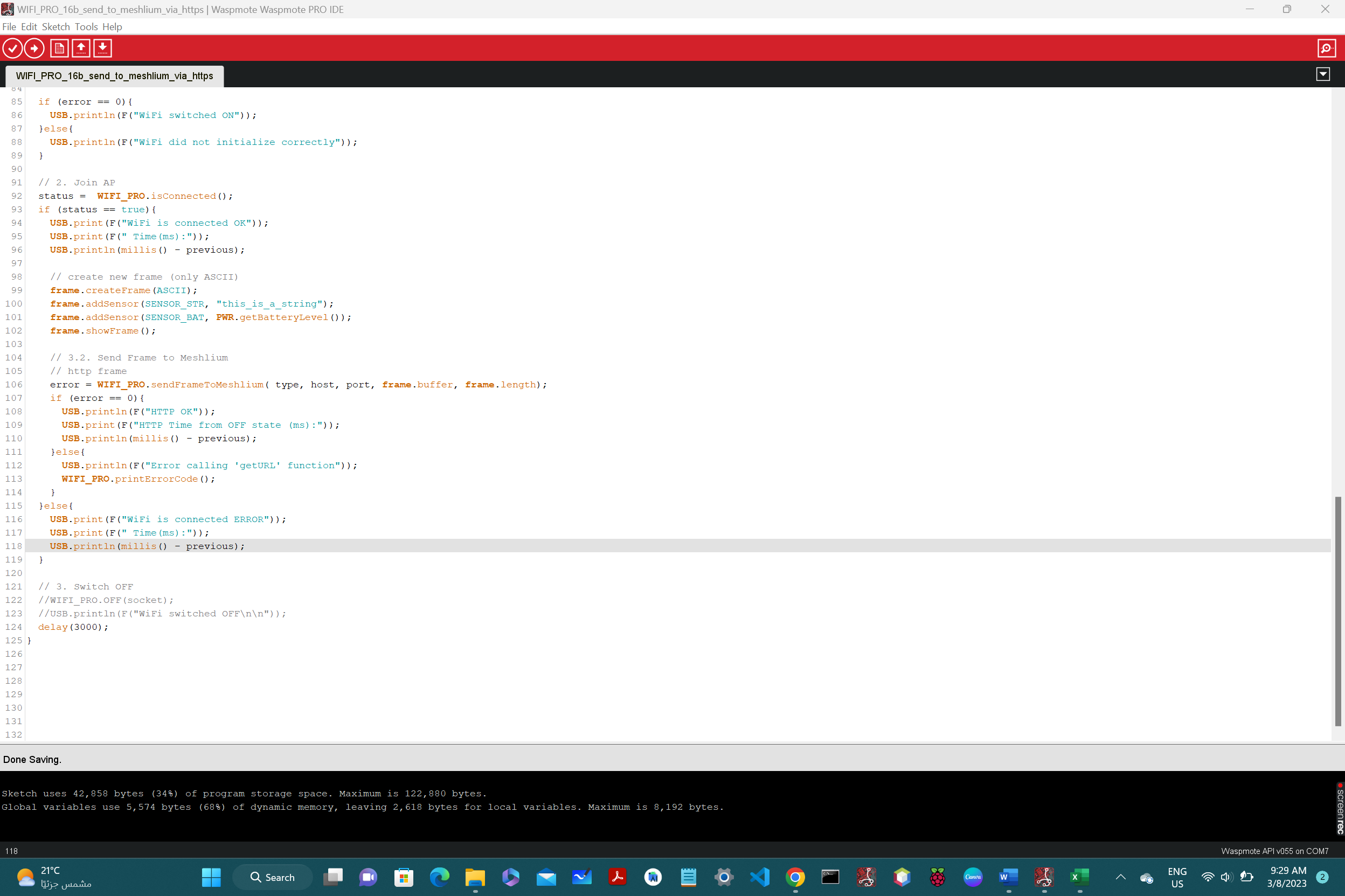
- Necessary variables are declared such as the socket to use, the type of connection (HTTPS), host IP address, and port number.

- A trusted CA certificate is declared and set using the 'setCA' function.

- The setup function initializes the Wi-Fi module and sets the trusted CA certificate.

- The loop function checks if the Wi-Fi is connected, creates a new frame with some sensor data, and sends it to the server using the 'sendFrameToMeshlium' function over an HTTPS connection.

- Finally, the code turns off the Wi-Fi module and waits three seconds before restarting the loop.



4: WIFI\_PRO\_01\_configure\_essid:

#include <WaspWIFI\_PRO.h>

char ESSID[] = "Robby\_1";

char PASSW[] = "nawaf123";

//char ESSID[] = "Students";

//char PASSW[] = "#UG2023";

//char ESSID[] = "D222";

//char PASSW[] = "$$123789$$";

uint8\_t socket = SOCKET0;

uint8\_t error;

uint8\_t status;

unsigned long previous;

void setup(){

USB.println(F("Start program"));

// 1. Switch ON the WiFi module

error = WIFI\_PRO.ON(socket);

if (error == 0){

USB.println(F("1. WiFi switched ON"));

}else{

USB.println(F("1. WiFi did not initialize correctly"));

}

// 2. Reset to default values

error = WIFI\_PRO.resetValues();

if (error == 0){

USB.println(F("2. WiFi reset to default"));

}else{

USB.println(F("2. WiFi reset to default ERROR"));

}

// 3. Set ESSID

error = WIFI\_PRO.setESSID(ESSID);

if (error == 0){

USB.println(F("3. WiFi set ESSID OK"));

}else{

USB.println(F("3. WiFi set ESSID ERROR"));

}

// 4. Set password key (It takes a while to generate the key)

// Authentication modes:

// OPEN: no security

// WEP64: WEP 64

// WEP128: WEP 128

// WPA: WPA-PSK with TKIP encryption

// WPA2: WPA2-PSK with TKIP or AES encryption

//////////////////////////////////////////////////

//error = WIFI\_PRO.setPassword(WPA2, PASSW);

error = WIFI\_PRO.setPassword(WPA2, PASSW);

if (error == 0){

USB.println(F("4. WiFi set AUTHKEY OK"));

}else{

USB.println(F("4. WiFi set AUTHKEY ERROR"));

}

// 5. Software Reset

// Parameters take effect following either a

// hardware or software reset

error = WIFI\_PRO.softReset();

if (error == 0){

USB.println(F("5. WiFi softReset OK"));

}else{

USB.println(F("5. WiFi softReset ERROR"));

}

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.println(F("Once the module is configured with ESSID"));

USB.println(F("and PASSWORD, the module will attempt to "));

USB.println(F("join the specified Access Point on power up"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

error = WIFI\_PRO.setTimeServer("time.google.com");

if (error == 0){

USB.println(F("time set succesfully"));

}else{

USB.println(F("Time setting ERROR"));

}

previous = millis();

}

void loop(){

if (WIFI\_PRO.isConnected() == true){

USB.print(F("WiFi is connected OK"));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

USB.println(F("\n\*\*\* Program stops \*\*\*"));

while(1){

}

}else{

USB.print(F("WiFi is connected ERROR"));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

}

}

Detailed explanation of the functionality of the code:

This code is configuring the WiFi module to connect to a specific wireless network with the ESSID "Robby\_1" and the password "nawaf123". In this code, the program is first initializing the WiFi module, resetting it to default values, setting the ESSID and password, and then doing a software reset to apply the changes. The last part of the code is checking if the WiFi module is connected to the wireless network or not. If it is connected, the program stops its execution. If not, it displays an error message. Overall, the code is setting up the WiFi module to connect to a specific wireless network and verifying the connection status.

Graphical user interface, text, application

Description automatically generated

5: WIFI\_PRO\_16a\_send\_to\_meshlium\_via\_http:

#include <WaspWIFI\_PRO.h>

#include <WaspFrame.h>

uint8\_t socket = SOCKET0;

// choose URL settings

char type[] = "http";

char host[] = "10.10.10.1";

char port[] = "80";

uint8\_t error;

uint8\_t status;

unsigned long previous;

char moteID[] = "node\_BME280";

void setup() {

USB.println(F("Start program"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.println(F("Once the module is set with one or more"));

USB.println(F("AP settings, it attempts to join the AP"));

USB.println(F("automatically once it is powered on"));

USB.println(F("Refer to example 'WIFI\_PRO\_01' to configure"));

USB.println(F("the WiFi module with proper settings"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

frame.setID(moteID);

}

void loop(){

previous = millis();

// 1. Switch ON

error = WIFI\_PRO.ON(socket);

if (error == 0){

USB.println(F("WiFi switched ON"));

}else{

USB.println(F("WiFi did not initialize correctly"));

}

// 2. Join AP

status = WIFI\_PRO.isConnected();

if (status == true){

USB.print(F("WiFi is connected OK"));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

// 3.1. Create a new Frame (only ASCII)

frame.createFrame(ASCII);

frame.addSensor(SENSOR\_STR, "nawaf");

frame.addSensor(SENSOR\_BAT, PWR.getBatteryLevel());

frame.showFrame(); /\* Print the frame here \*/

// 3.2. Send Frame to Meshlium

error = WIFI\_PRO.sendFrameToMeshlium( type, host, port, frame.buffer, frame.length);

if (error == 0){

USB.println(F("HTTP OK"));

USB.print(F("HTTP Time from OFF state (ms):"));

USB.println(millis()-previous);

}else{

USB.println(F("Error calling 'getURL' function"));

WIFI\_PRO.printErrorCode();

}

}else{

USB.print(F("WiFi is connected ERROR"));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

}

// 3. Switch OFF

//WIFI\_PRO.OFF(socket);

//USB.println(F("WiFi switched OFF\n\n"));

delay(3000);

}

Detailed explanation of the functionality of the code:

This code is used to send data from a Waspmote device to a Meshlium Gateway using HTTP protocol. The code initializes the WiFi module, connects to an Access Point, creates a Frame with sensor data, sends the frame to the Meshlium gateway and then disconnects from the Access Point.

The code first includes two header files: `WaspWIFI\_PRO.h` and `WaspFrame.h`. The `WaspWIFI\_PRO.h` header file contains the necessary functions to communicate with the WiFi module, while the `WaspFrame.h` header file contains the functions to create and manipulate wireless sensor frames.

The `setup()` function initializes the Waspmote board and sets the moteID for the frame. The `loop()` function is run repeatedly and consists of the following steps:

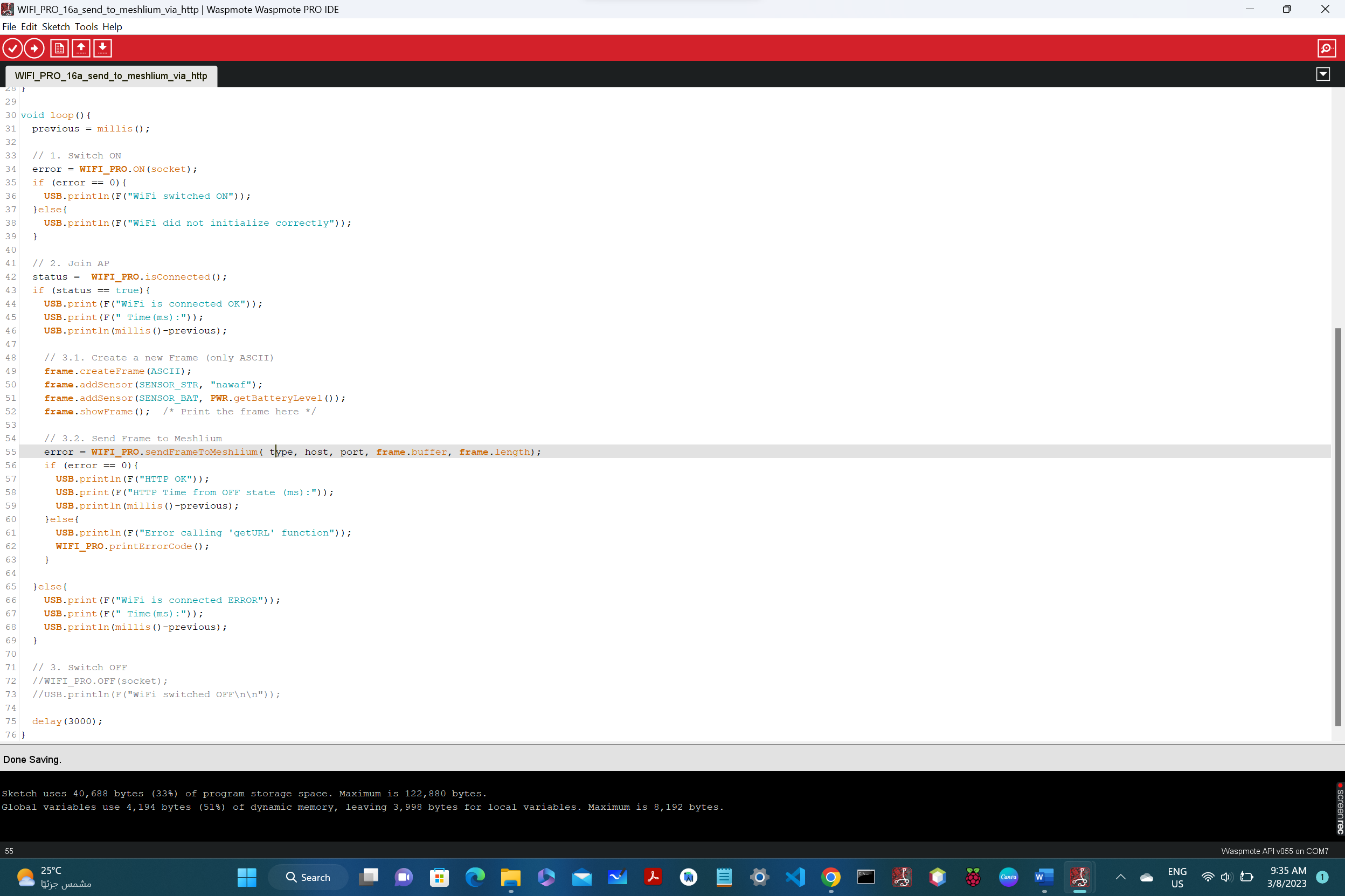
1. Switch ON WiFi: The code turns on the WiFi module by calling the `WIFI\_PRO.ON()` function. The return value of `WIFI\_PRO.ON()` function will be zero on success and non-zero on failure.

2. Join Access Point: The code checks whether the WiFi module has connected to the Access Point by calling the `WIFI\_PRO.isConnected()` function. The return value of `WIFI\_PRO.isConnected()` function will be true if the device is connected to the Access Point and false otherwise.

3. Create and send Frame: Once the WiFi module is connected, a new Frame is created using the `WaspFrame.createFrame()` function. In this example, the frame contains two sensor values: "nawaf" and battery level. After creating the frame, the code calls the `WIFI\_PRO.sendFrameToMeshlium()` function to send the frame to the Meshlium gateway.

4. Switch OFF WiFi: Finally, the code turns off the WiFi module by calling `WIFI\_PRO.OFF()` function.

The code also prints debugging information using the `USB.println()` function to the Serial Monitor.



6: WIFI\_PRO\_002\_JOIN:

#include <WaspWIFI\_PRO.h>

#include <WaspSensorEvent\_v30.h> /\* BME280, Events.ON()\*/

uint8\_t socket = SOCKET0;

uint8\_t error;

uint8\_t status;

unsigned long previous;

float temp,humd,pres;

void setup() {

USB.ON();

USB.println(F("Start program"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.println(F("Once the module is set with one or more"));

USB.println(F("AP settings, it attempts to join the AP"));

USB.println(F("automatically once it is powered on"));

USB.println(F("Refer to example 'WIFI\_PRO\_01' to configure"));

USB.println(F("the WiFi module with proper settings"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

error = WIFI\_PRO.ON(socket);

if (error == 0){

USB.println(F("1. WiFi switched ON"));

}else{

USB.println(F("1. WiFi did not initialize correctly"));

}

error = WIFI\_PRO.setTimeServer("time.google.com");

if (error == 0){

USB.println(F("1.1 Time set succesfully"));

}else{

USB.println(F("Time setting ERROR"));

}

previous = millis();

}

/\*\*\*\*\*\*\*\*\*\* LOOP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop(){

// 2. Join AP

status = WIFI\_PRO.isConnected();

if (status == true){

USB.print(F("2. WiFi is connected OK."));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

error = WIFI\_PRO.ping("www.google.com");

if (error == 0){

USB.print(F("3. PING OK. Round Trip Time(ms)="));

USB.println( WIFI\_PRO.\_rtt, DEC );

}else{

USB.println(F("3. Error calling 'ping' function"));

}

}else{

USB.print(F("2. WiFi is connected ERROR."));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

}

/\*\*\*\*\*\*\*\*\* REAM BME280 SENSOR \*\*\*\*\*\*\*\*\*\*\*\*/

Events.ON();

temp = Events.getTemperature();

humd = Events.getHumidity();

pres = Events.getPressure();

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\* BME380 \*\*\*\*\*\*\*\*\*"));

USB.print(F("Temperature: "));

USB.print(temp);

USB.println(F(" Celsius"));

USB.print(F("Humidity: "));

USB.print(humd);

USB.println(F(" %"));

USB.print(F("Pressure: "));

USB.print(pres);

USB.println(F(" Pa"));

USB.println();

delay(10000);

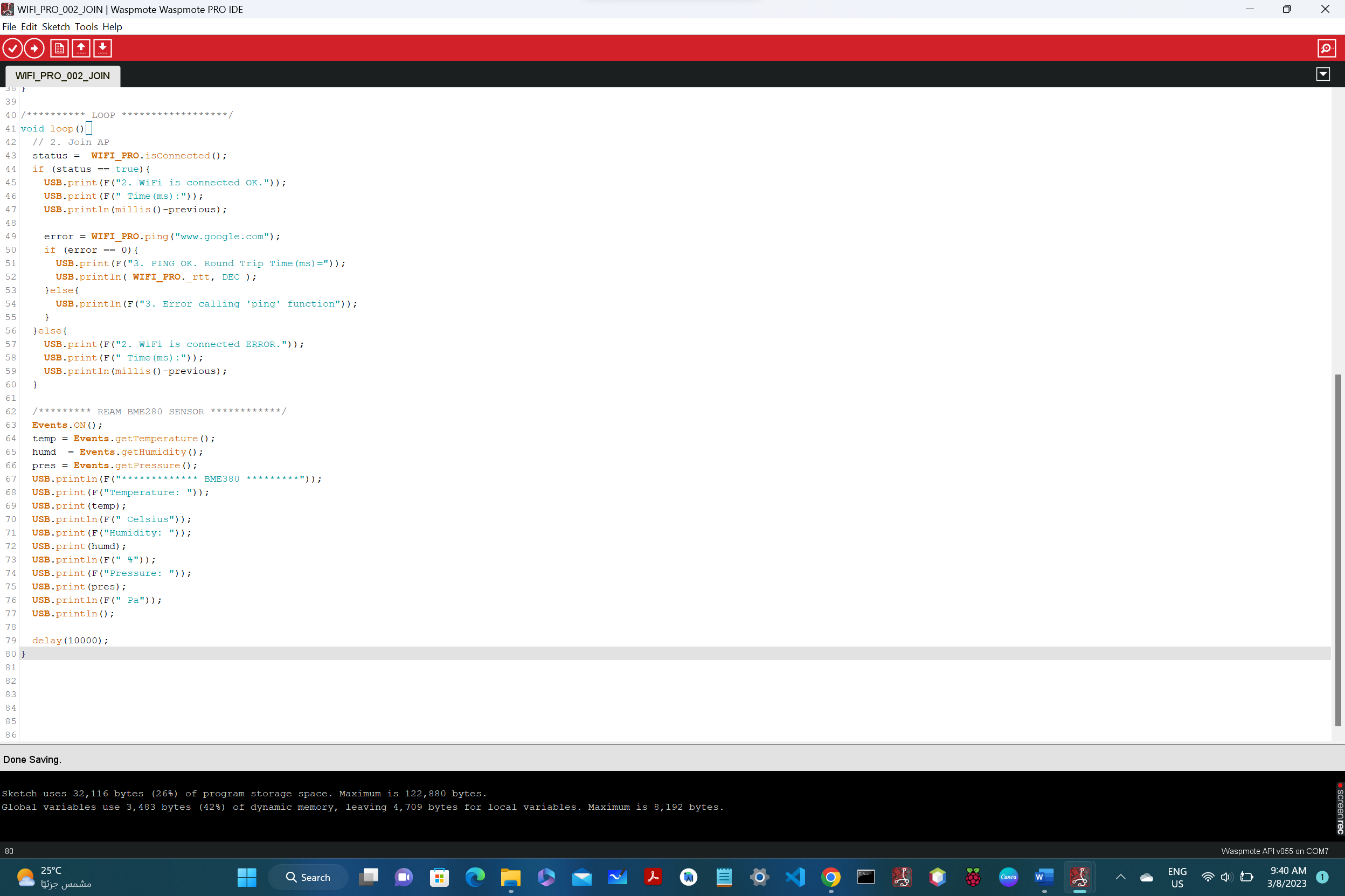
}

Detailed explanation of the functionality of the code:

This code is used for setting up and joining a Wi-Fi network using a Waspmote device. The code uses the Waspmote libraries for Wi-Fi and sensor event handling.

In the setup() function, USB communication is turned on and the program's purpose is displayed. The Wi-Fi module is turned on and checked for initialization. A time server is set to accurately set the time for the device.

In the loop() function, the status of Wi-Fi network connection is checked. If the device is connected to the network, it pings the Google server and displays the round trip time. If not, it displays an error message. It also reads BME280 sensor values and displays them over USB communication. Lastly, there is a 10 seconds delay before the loop repeats.



7: read\_temp\_sensor\_BME280:

#include <WaspSensorEvent\_v30.h>

float temp;

float humd;

float pres;

float value;

void setup(){

USB.ON();

USB.println(F("Read from expansion board, EVENTS"));

Events.ON();

}

void loop(){

temp = Events.getTemperature();

humd = Events.getHumidity();

pres = Events.getPressure();

USB.println("-----------------------------");

USB.print("Temperature: ");

USB.printFloat(temp, 2);

USB.println(F(" Celsius"));

USB.print("Humidity: ");

USB.printFloat(humd, 1);

USB.println(F(" %"));

USB.print("Pressure: ");

USB.printFloat(pres, 2);

USB.println(F(" Pa"));

USB.println("-----------------------------");

/\*

USB.println(F("Going into deep sleep mode"));

PWR.deepSleep("00:00:00:10", RTC\_OFFSET, RTC\_ALM1\_MODE1, ALL\_OFF);

USB.ON();

USB.println(F("wake up\n"));

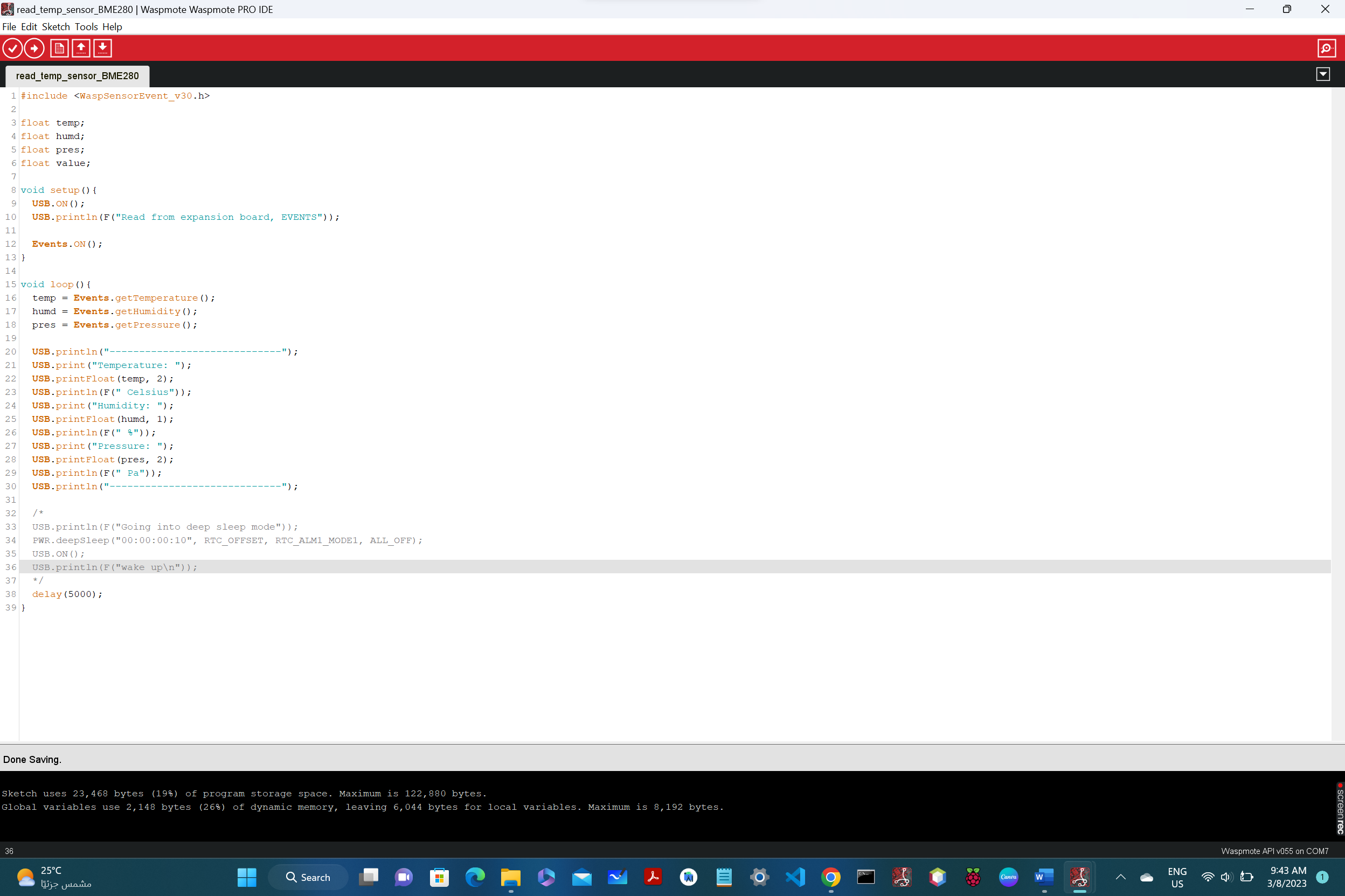
\*/

delay(5000);

}

Detailed explanation of the functionality of the code:

This piece of code reads the data from a BME280 sensor using a Waspmote device. The code creates four float variables to store temperature, humidity, pressure, and a general value. In the setup function, the USB port is turned on and a message is printed to the console. The Events library is also turned on. In the main loop, the Events library is used to get the current temperature, humidity, and pressure values from the sensor. The values are then printed to the console using the USB port. Finally, a delay of 5 seconds is added before the loop starts over again. It is worth noting that the code includes commented-out code for putting the device into deep sleep mode.



8: \_802\_16\_set\_time\_from\_meshlium:

#include <WaspXBee802.h>

/\* IoT LAB Robby\_1 MAC address MESHLIUM \*/

char MESHLIUM\_ADDRESS[] = "0013A20041678A0E";

uint8\_t socket = SOCKET0;

uint8\_t error;

void setup(){

USB.ON();

RTC.ON();

xbee802.ON(socket);

}

void loop(){

USB.println(F("Setting RTC time from Meshlium settings"));

error = xbee802.setRTCfromMeshlium(MESHLIUM\_ADDRESS);

if( error == 0 ){

USB.print(F("SET RTC ok. "));

}else{

USB.print(F("SET RTC error. "));

}

USB.print(F("RTC Time:"));

USB.println(RTC.getTime());

delay(9000);

}

Detailed explanation of the functionality of the code:

This code is using the WaspXBee802 library in order to set the real-time clock (RTC) on a Waspmote device by getting the time from a Meshlium device via a wireless XBee802 connection.

The code creates a variable "socket" with the integer value 0, and initializes the XBee802 module, the USB port, and the RTC on the Waspmote device in the setup function.

In the loop, it prints the message "Setting RTC time from Meshlium settings" to the USB port, then calls xbee802.setRTCfromMeshlium() with the MAC address of the Meshlium device.

The function xbee802.setRTCfromMeshlium() sends a packet to the Meshlium device requesting its current time, and then sets the RTC on the Waspmote device to that time. The function returns an error code (0 for success) which is stored in the "error" variable.

After setting the RTC, the code prints the current time on the Waspmote device (which should now be synced with the Meshlium device) to the USB port. It then adds a delay of 9000 milliseconds (or 9 seconds) before looping again to repeat the process.

Graphical user interface, text, application

Description automatically generated

9: \_802\_get\_set\_Freq\_Channel\_PAN\_ID:

#include <WaspXBee802.h>

#include <WaspSensorCities\_PRO.h>

char CHANNEL[]="C";

uint8\_t error;

uint8\_t panID=3332;

void setup(){

xbee802.ON();

USB.println(xbee802.getMacMode(), DEC);

USB.print("---> My MAC ADDRESS: ");

xbee802.getOwnMacHigh();

xbee802.getOwnMacLow();

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacHigh[i]);

}

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacLow[i]);

}

USB.println();

xbee802.getChannel();

USB.print(F("---> INITIAL xbee802.channel: "));

USB.printHex(xbee802.channel);

USB.println();

xbee802.setChannel(0xC);

USB.print(F("---> xbee802.channel: "));

USB.printHex(xbee802.channel); // CORRECT, it is set in channel C

USB.println();

//error =

//xbee802.setPAN(panID);

/\*

if(error!=0){

println("SetPAN error...");

}

\*/

xbee802.getPAN();

USB.print(F("---> xbee802.getPAN: "));

for (int i=0;i<8;i++){

USB.print(xbee802.PAN\_ID[i], HEX);

}

//USB.printHex(xbee802.PAN\_ID);

USB.println();

xbee802.getNodeIdentifier();

USB.print(F("---> node ID: "));

USB.println(xbee802.nodeID);

}

void loop(){

}

Detailed explanation of the functionality of the code:

This code defines a function `\_802\_get\_set\_Freq\_Channel\_PAN\_ID` that interacts with XBee radio module. The purpose of this function is to set the Channel and PAN ID of the XBee module to a pre-defined value (Channel `C` and PAN ID `3332`).

This function uses the `WaspXBee802.h` library which provides an interface for sending and receiving data through the XBee radio module.

The `setup()` function is responsible for initializing the XBee module by turning it on, configuring channel and PAN ID of the module, and retrieving its MAC address.

`xbee802.ON()` turns the XBee module on. `USB.println(xbee802.getMacMode(), DEC)` prints the current MAC addressing mode used by the XBee module in decimal format. The module's MAC address is retrieved by calling `xbee802.getOwnMacHigh()` and `xbee802.getOwnMacLow()` methods which are used to read the address from the module's registers.

The current channel number used by the module is retrieved by calling `xbee802.getChannel()`, and it is printed to the console using `USB.printHex(xbee802.channel)`.

The channel of the XBee module is then set to channel `C` using `xbee802.setChannel(0xC)`, and the new channel number is printed to the console.

The PAN ID of the XBee module is set to `3332` using `xbee802.setPAN(panID)` which is commented out in this code. The current PAN ID of the module is retrieved by calling `xbee802.getPAN()`, and it is printed to the console using `USB.print(xbee802.PAN\_ID[i], HEX)`.

Finally, `xbee802.getNodeIdentifier()` retrieves the node ID of the XBee module, and it is printed to the console using `USB.println(xbee802.nodeID)`.

The `loop()` function is empty and is not used in this code.

Graphical user interface, text, application

Description automatically generated

10: \_1\_CLEAR\_NODE:

void setup(){

USB.ON();

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\* MOTE IS EMPTY \*\*\*\*\*\*\*\*\*\*\*\*\*"));

}

void loop(){

}

Detailed explanation of the functionality of the code:

The code defines a function called `\_1\_CLEAR\_NODE` which appears to be a label or a placeholder for a specific action or functionality. However, there is no implementation mentioned within this code snippet.

Additionally, this code contains two other functions; the first one is called `setup()` which is executed once when the device is powered on, and the second function is called `loop()` which is executed repeatedly in a loop.

Inside the `setup()` function, it turns on the USB communication port and prints a message `"\*\*\*\*\*\*\*\*\*\*\*\* MOTE IS EMPTY \*\*\*\*\*\*\*\*\*\*\*\*\*"` on the serial monitor.

The `loop()` function is empty, meaning it does not contain any instructions to be executed.

Overall, this code doesn't seem to have a specific functionality, but it sets up the USB communication port and indicates that the "MOTE is empty".

Graphical user interface, text, application

Description automatically generated

11: \_3ple\_sensor\_reading:

#include <WaspSensorGas\_Pro.h>

/\*

\* Waspmote OEM. Possibilities for this sensor:

\* - CENTRAL SOCKET

\* P&S! Possibilities for this sensor:

\* - SOCKET\_E

\*/

bmeGasesSensor bme;

uint8\_t socket = SOCKET0;

float temperature; // Stores the temperature in ºC

float humidity; // Stores the realitve humidity in %RH

float pressure; // Stores the pressure in Pa

void setup()

{

USB.ON();

USB.println(F("Temperature, humidity and pressure sensor example"));

}

void loop()

{

/\* 1. Turn on the sensor \*/

bme.ON();

/\* 2. Read sensors \*/

temperature = bme.getTemperature();

humidity = bme.getHumidity();

pressure = bme.getPressure();

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.print(F("Temperature: "));

USB.print(temperature);

USB.println(F(" Celsius degrees"));

USB.print(F("RH: "));

USB.print(humidity);

USB.println(F(" %"));

USB.print(F("Pressure: "));

USB.print(pressure);

USB.println(F(" Pa"));

/\* 3. Turn off the sensor \*/

bme.OFF();

/\* 3. Sleep

\* After 3 minutes, Waspmote wakes up thanks to the RTC Alarm \*/

USB.println(F("Go to deep sleep mode..."));

PWR.deepSleep("00:00:00:05", RTC\_OFFSET, RTC\_ALM1\_MODE1, ALL\_OFF);

USB.ON();

USB.println(F("Wake up!!\r\n"));

}

Detailed explanation of the functionality of the code:

This code is used to read temperature, humidity, and pressure using a BME gases sensor in Waspmote. The first thing the code does is to include the WaspSensorGas\_Pro library.

After that, it defines some global variables such as socket, temperature, humidity, and pressure.

The setup() function just initializes the USB communication for debugging purposes.

The loop() function is where the main sensor reading happens. First, the bme.ON() function is called to turn on the sensor.

Then, the temperature, humidity, and pressure are read using the bme.getTemperature(), bme.getHumidity(), and bme.getPressure() functions, respectively.

The results are printed using the USB.println() function for debugging purposes.

After that, the bme.OFF() function is called to turn off the sensor, and the Waspmote goes to deep sleep for three minutes using the PWR.deepSleep() function.

Finally, the USB communication is turned back on for debugging, and the Waspmote wakes up again.

Graphical user interface, text, application

Description automatically generated

12: \_802\_15\_4\_TX\_MINIMAL:

#include <WaspXBee802.h>

#include <WaspFrame.h>

char RX\_ADDRESS[] = "000000000000FFFF"; /\* BROADCAST \*/

//char RX\_ADDRESS[] = "0013A20041678A0E"; /\* AP MEshlium MAC \*/

//char RX\_ADDRESS[] = "0013A2004149DA14"; //XBEE S1 from IoT lab

char NODE\_ID[] = "TX\_node\_min";

uint8\_t error;

void setup(){

USB.ON();

USB.println(F("\n\*\*\*\*\*\*\* 802.15.4 SEND (TX) MINIMAL \*\*\*\*\*\*\n"));

xbee802.ON();

USB.print("---> My MAC ADDRESS: ");

xbee802.getOwnMacHigh();

xbee802.getOwnMacLow();

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacHigh[i]);

}

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacLow[i]);

}

USB.println();

xbee802.getChannel();

USB.print(F("---> xbee802.channel: "));

USB.printHex(xbee802.channel); // CORRECT, it is set in channel C

USB.println();

xbee802.setNodeIdentifier(NODE\_ID);

xbee802.getNodeIdentifier();

USB.print(F("---> node ID: "));

USB.println(xbee802.nodeID); /\* Should print the whole array \*/

if(xbee802.setOwnNetAddress("1212") == 0){

USB.println(F("---> setOwnNetAddress(\"1212\") success"));

} else{

USB.println(F("---> setOwnNetAddress(\"1212\") ERROR..."));

}

}

void loop(){

delay(3000); // it goes too fast

USB.println(F("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* LOOP STARTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

USB.println(F("-------- 1. Create ASCII frame ----------"));

frame.createFrame(ASCII);

frame.setFrameSize(125);

frame.addSensor(SENSOR\_BAT, PWR.getBatteryLevel());

error =

xbee802.send(RX\_ADDRESS, frame.buffer, frame.length);

if( error == 0 ){ // check TX flag

USB.println(F("---> frame sent ok"));

}

else{

USB.println(F("---> OOPS..., frame sent error"));

Utils.blinkRedLED();

}

int delayTime = 3; /\* seconds \*/

USB.print(F("\n\*\*\*\*\* LOOP END, Wait for "));

USB.print(delayTime);

USB.print(F(" sec \*\*\*\*\*\*\*\*\*\n"));

delay(delayTime \* 1000); /\* milliseconds \*/

}

Detailed explanation of the functionality of the code:

This code is designed to send data wirelessly using the XBee module in 802.15.4 protocol. The code first sets the RX address to "000000000000FFFF" for broadcast, and includes two other options, but currently commented out. Next, it sets the NODE\_ID, which identifies the transmitting module.

In the setup() function, the program turns on USB communication, turns on the XBee module, and prints the module's MAC address and channel. It then sets the Node Identifier and the module's network address to "1212", with console output indicating success or failure of the latter.

The loop() function begins by creating an ASCII frame of size 125, and adds data from the battery level sensor to it. The frame is then sent using the send() method with the previously set RX\_ADDRESS. If the frame is sent successfully, console output confirms this. Finally, there is a delay of three seconds before the loop starts again.

Text

Description automatically generated

13: \_802\_15\_4\_TX\_WITH\_BME280:

#include <WaspXBee802.h>

#include <WaspFrame.h>

#include <WaspSensorEvent\_v30.h>

/\* https://github.com/NawafAlzahem/Capstone-Project.git \*/

/\* BME280 triple sensor \*/

float temp;

float humd;

float pres;

float value;

// TODO: ONLY BROADCAST SEEMS TO WORK

char RX\_ADDRESS[] = "000000000000FFFF"; /\* BROADCAST \*/

//char RX\_ADDRESS[] = "0013A20041678A0E"; /\* AP MEshlium MAC \*/

//char RX\_ADDRESS[] = "0013A2004149DA14"; //XBEE S1 from IoT lab

//char RX\_ADDRESS[] = "0013A2004149DA23";

char MY\_NET\_ADDRESS[]="2112";

char RX\_NET\_ADDRESS[]="1221";

char MESHLIUM\_NET\_ADDRESS[]="1111";

/\* In 802.15.4 you can use either full 8 HEX standard MAC

\* Address, or you can use the 2 HEX NET address.

\* The receiver will adapt automatically.

\* 0= use NET ADDRESS, 1 = USE MAC \*/

uint8\_t ADDRESS\_TYPE = 1;

char NODE\_ID[] = "TX\_WITH\_BME280";

char node\_data[20];

uint8\_t error;

int counter = 0;

int delayTime;

uint8\_t api\_mode = 1; /\* 0-2 \*/

uint8\_t mac\_mode = 0; /\* 0-3 \*/

void setup(){

USB.ON();

USB.println(F("\n\*\*\*\*\*\*\* 802.15.4 WITH BME280 (TX) \*\*\*\*\*\*\n"));

xbee802.ON();

Events.ON(); /\* expansion board for sensors \*/

xbee802.setNodeIdentifier(NODE\_ID);

xbee802.getNodeIdentifier();

USB.print(F("---> node ID: "));

USB.println(xbee802.nodeID); /\* Should print the whole array \*/

USB.print(F("---> API mode: "));

USB.println(api\_mode, DEC);

/\* Set API mode accordingly \*/

xbee802.setAPI(api\_mode); /\* WHEN SET OTHERWISE, IT IS NOT WORKING !! \*/

/\* to select at any time if the modules are to use

\* a totally compatible heading format \*/

xbee802.setMacMode(mac\_mode); /\* 0-3 \*/

USB.print(F("---> MAC mode: "));

USB.println(xbee802.getMacMode(), DEC);

USB.print("---> My MAC ADDRESS: ");

xbee802.getOwnMacHigh();

xbee802.getOwnMacLow();

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacHigh[i]);

}

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacLow[i]);

}

USB.println();

xbee802.getChannel();

USB.print(F("---> xbee802.channel: "));

USB.printHex(xbee802.channel); // CORRECT, it is set in channel C

USB.println();

USB.print(F("---> setOwnNetAddress("));

USB.print(MY\_NET\_ADDRESS);

if(xbee802.setOwnNetAddress(MY\_NET\_ADDRESS) == 0){

USB.print(F(") SUCCESSFUL..."));

} else{

USB.print(F(") ERROR!..."));

}

USB.println();

}

void loop(){

USB.println(F("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* LOOP STARTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

temp = Events.getTemperature();

humd = Events.getHumidity();

pres = Events.getPressure();

USB.println("-------- BME280 ---------");

USB.print("Temperature: ");

USB.printFloat(temp, 2);

USB.println(F(" Celsius"));

USB.print("Humidity: ");

USB.printFloat(humd, 1);

USB.println(F(" %"));

USB.print("Pressure: ");

USB.printFloat(pres, 2);

USB.println(F(" Pa"));

USB.println("-----------------------------\n");

counter++;

sprintf(node\_data,"Packet No:%d",counter);

USB.println(F("-------- 1. Create ASCII frame ----------"));

frame.createFrame(ASCII, NODE\_ID); /\* Set the node ID INSIDE the frame \*/

frame.setFrameSize(125);

/\* EXAMPLE\_FRAME

TIMEOUT\_FRAME

EVENT\_FRAME

ALARM\_FRAME

SERVICE1\_FRAME

SERVICE2\_FRAME \*/

frame.setFrameType(EVENT\_FRAME);

frame.addSensor(SENSOR\_STR, node\_data);

frame.addSensor(SENSOR\_BAT, PWR.getBatteryLevel());

frame.addSensor(SENSOR\_TCA, temp); /\* look into data\_frame\_guide.pdf \*/

frame.addSensor(SENSOR\_HUMA, humd);

frame.addSensor(SENSOR\_PA, pres);

USB.println(F("---> Frame 2 send: ")); frame.showFrame();

if(ADDRESS\_TYPE == 0){ /\* 0 or 1 \*/

error = xbee802.send(RX\_NET\_ADDRESS, frame.buffer, frame.length);

}else{

error = xbee802.send(RX\_ADDRESS, frame.buffer, frame.length);

}

if( error == 0 ){ // check TX flag

USB.println(F("---> frame sent ok"));

Utils.blinkGreenLED();

/\* it should deteriorate with distance \*/

USB.print(F("---> RSSI(dBm): "));

USB.println(xbee802.\_rssi);

/\* Receive XBee packet (wait message for 2 seconds) \*/

error = xbee802.receivePacketTimeout( 4000 );

if( error == 0 ){

USB.println(F("------- 1.1 Responce packet received ---------"));

/\* define the variable ad hoc \*/

char data\_in[xbee802.\_length]; // assign the length correct value

/\* Show data stored in '\_payload' buffer indicated by '\_length' \*/

USB.print(F("---> Received message: "));

for(int i=0;i<xbee802.\_length;i++){

USB.print(xbee802.\_payload[i]);

}

USB.println();

}

} else {

USB.println(F("---> OOPS..., frame sent error"));

Utils.blinkRedLED();

}

delayTime = 5; /\* seconds \*/

USB.print(F("\n\*\*\*\*\* LOOP END, Wait for "));

USB.print(delayTime);

USB.print(F(" sec \*\*\*\*\*\*\*\*\*\n"));

delay(delayTime \* 1000); /\* milliseconds \*/

}

Detailed explanation of the functionality of the code:

This code enables the BME280 sensor to send data wirelessly to a receiving device using the 802.15.4 protocol.

The code sets up the necessary parameters for the sensor and the wireless module, including the node identifier, MAC and NET addresses, and API and MAC modes. It then enters an endless loop that reads data from the BME280 sensor, formats it into an ASCII frame, and sends it wirelessly using the 802.15.4 protocol. The code also contains commands for receiving response packets back from the receiving device, as well as for displaying the data on the terminal.

The code can be configured to broadcast the data to all devices within range, or to send it to a specific MAC address using the 802.15.4 protocol.

Text

Description automatically generated

14: WIFI\_PRO\_01\_configure\_essid\_START\_THIS:

#include <WaspWIFI\_PRO.h>

uint8\_t socket = SOCKET1;

// WiFi AP settings (CHANGE TO USER'S AP)

char ESSID[] = "Robby\_1";

char PASSW[] = "nawaf123";

uint8\_t error;

uint8\_t status;

unsigned long previous;

void setup() {

USB.println(F("Start program"));

// 1. Switch ON the WiFi module

error = WIFI\_PRO.ON(socket);

if (error == 0){

USB.println(F("1. WiFi switched ON"));

} else {

USB.println(F("1. WiFi did not initialize correctly"));

}

// 2. Reset to default values

error = WIFI\_PRO.resetValues();

if (error == 0){

USB.println(F("2. WiFi reset to default"));

} else {

USB.println(F("2. WiFi reset to default ERROR"));

}

// 3. Set ESSID

error = WIFI\_PRO.setESSID(ESSID);

if (error == 0) {

USB.println(F("3. WiFi set ESSID OK"));

} else {

USB.println(F("3. WiFi set ESSID ERROR"));

}

/\* 4. Set password key (It takes a while to generate the key)

\* Authentication modes:

\* OPEN: no security

\* WEP64: WEP 64

\* WEP128: WEP 128

\* WPA: WPA-PSK with TKIP encryption

\* WPA2: WPA2-PSK with TKIP or AES encryption \*/

error = WIFI\_PRO.setPassword(WPA2, PASSW);

if (error == 0){

USB.println(F("4. WiFi set AUTHKEY OK"));

} else {

USB.println(F("4. WiFi set AUTHKEY ERROR"));

}

/\* 5. Software Reset

\* Parameters take effect following either a

\* hardware or software reset \*/

error = WIFI\_PRO.softReset();

if (error == 0) {

USB.println(F("5. WiFi softReset OK"));

} else {

USB.println(F("5. WiFi softReset ERROR"));

}

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.println(F("Once the module is configured with ESSID"));

USB.println(F("and PASSWORD, the module will attempt to "));

USB.println(F("join the specified Access Point on power up"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

previous = millis();

}

void loop() {

if (WIFI\_PRO.isConnected() == true) {

USB.print(F("WiFi is connected OK"));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

USB.println(F("\n\*\*\* Program stops \*\*\*"));

while(1) {

}

} else {

USB.print(F("WiFi is connected ERROR"));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

}

}

Detailed explanation of the functionality of the code:

This code configures a Waspmote device to connect to a WiFi network using the WiFi Pro module. Here are the main steps:

1. The `WaspWIFI\_PRO` library is included to allow communication with the WiFi module.

2. A variable `socket` is defined to specify which socket to use for communication with the module.

3. The SSID and password of the WiFi network to be connected to are defined as `ESSID` and `PASSW` respectively.

4. In the `setup()` function, the WiFi module is turned on by calling `WIFI\_PRO.ON(socket)`. If no errors occur, the program prints "WiFi switched ON".

5. The `WIFI\_PRO.resetValues()` function is called to reset the WiFi module to its default values.

6. The `WIFI\_PRO.setESSID()` function is called with the `ESSID` variable as its argument to set the WiFi network's SSID.

7. The `WIFI\_PRO.setPassword()` function is called with the `PASSW` variable and the authentication mode `WPA2` to set the WiFi network's password.

8. The `WIFI\_PRO.softReset()` function is called to apply the changes made in the previous steps.

9. The program prints some messages to the console to inform the user that the WiFi module has been configured.

10. In the `loop()` function, the `WIFI\_PRO.isConnected()` function is called to check if the module is connected to a WiFi network. If it is, the program prints a success message and stops running. If not, an error message is printed instead.

The comments in the code provide helpful explanations for each step.

Graphical user interface, text, application

Description automatically generated

15: WIFI\_PRO\_02\_JOIN\_ONLY\_WITH\_TIME\_SET:

#include <WaspWIFI\_PRO.h>

#include <WaspSensorEvent\_v30.h> /\* BME280, Events.ON()\*/

uint8\_t socket = SOCKET0;

uint8\_t error;

uint8\_t status;

unsigned long previous;

void setup() {

USB.ON();

USB.println(F("Start program"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.println(F("Once the module is set with one or more"));

USB.println(F("AP settings, it attempts to join the AP"));

USB.println(F("automatically once it is powered on"));

USB.println(F("Refer to example 'WIFI\_PRO\_01' to configure"));

USB.println(F("the WiFi module with proper settings"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

error = WIFI\_PRO.ON(socket);

if (error == 0){

USB.println(F("1. WiFi switched ON"));

}else{

USB.println(F("1. WiFi did not initialize correctly"));

}

error = WIFI\_PRO.setTimeServer("time.google.com");

if (error == 0){

USB.println(F("1.1 Time set succesfully"));

}else{

USB.println(F("Time setting ERROR"));

}

previous = millis();

}

/\*\*\*\*\*\*\*\*\*\* LOOP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void loop(){

// 2. Join AP

status = WIFI\_PRO.isConnected();

if (status == true){

USB.print(F("2. WiFi is connected OK."));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

error = WIFI\_PRO.ping("www.google.com");

if (error == 0){

USB.print(F("3. PING OK. Round Trip Time(ms)="));

USB.println( WIFI\_PRO.\_rtt, DEC );

}else{

USB.println(F("3. Error calling 'ping' function"));

}

}else{

USB.print(F("2. WiFi is connected ERROR."));

USB.print(F(" Time(ms):"));

USB.println(millis()-previous);

}

delay(10000);

}

Detailed explanation of the functionality of the code:

This code attempts to join a previously-configured Access Point (AP) automatically once it is powered on. It sets the time server to "time.google.com" using the setTimeServer() function. The loop() function checks if the module is connected to the AP and displays a message indicating whether the connection was successful or not along with the amount of time elapsed. It also pings "www.google.com" and displays the round trip time if the ping was successful. If the module is not connected, it displays an error message. The loop() function is executed repeatedly after a delay of 10 seconds using the delay() function.

Text

Description automatically generated

16: SMART\_CITIES\_SCP\_v30\_05\_Temp\_hum\_pres\_sens:

#include <WaspSensorCities\_PRO.h>

#include <WaspXBee802.h>

/\*

Waspmote OEM. Possibilities for this sensor:

- SOCKET\_1

- SOCKET\_2

- SOCKET\_3

- SOCKET\_4

- SOCKET\_5

P&S! Possibilities for this sensor:

- SOCKET\_A

- SOCKET\_B

- SOCKET\_C

- SOCKET\_E

- SOCKET\_F

\*/

/\* MAKE SURE YOU FIND THE CORRECT SOCKET! \*/

bmeCitiesSensor bme(SOCKET\_E);

float temperature; // Stores the temperature in ºC

float humidity; // Stores the realitve humidity in %RH

float pressure; // Stores the pressure in Pa

uint8\_t error;

void setup(){

USB.println(F("\*\*\*\* Temperature, humidity and pressure sensor example \*\*\*\*"));

RTC.ON();

xbee802.ON();

USB.print(F("RTC INITIAL Time:"));

USB.println(RTC.getTime());

// Show '\_serial\_id' stored by the API when powering up

USB.print(F("Global variable '\_serial\_id':"));

for (int i=0;i<8;i++){

USB.printHex(\_serial\_id[i]);

}

USB.println();

}

void loop(){

bme.ON();

temperature = bme.getTemperature();

humidity = bme.getHumidity();

pressure = bme.getPressure();

USB.println(F("\*\*\*\*\*\*\*\*\*\*\* TEMP-HUM-PRESS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.print(F("Temperature: "));

USB.printFloat(temperature, 2);

USB.println(F(" Celsius degrees"));

USB.print(F("RH: "));

USB.printFloat(humidity, 2);

USB.println(F(" %"));

USB.print(F("Pressure: "));

USB.printFloat(pressure, 2);

USB.println(F(" Pa"));

bme.OFF();

delay(4000);

/\*

// 4. deepSleep for 10 sec

USB.println(F("Go to deep sleep mode..."));

PWR.deepSleep("00:00:00:10", RTC\_OFFSET, RTC\_ALM1\_MODE1, ALL\_OFF);

USB.println(F("Wake up!!\r\n"));

\*/

}

Detailed explanation of the functionality of the code:

This code is for measuring the temperature, humidity, and pressure of the environment using the BME280 sensor. The code is designed to be used with the Waspmote platform, which is a low-power platform for the Internet of Things (IoT) applications.

The code includes two libraries: "WaspSensorCities\_PRO.h" and "WaspXBee802.h". These libraries offer an interface for working with the BME280 sensor and the ZigBee communication module.

The code starts by specifying the socket where the BME280 sensor is connected. Then, it initializes the sensor and reads the temperature, humidity, and pressure values. Finally, it prints these values to the USB port, waits for 4 seconds, and repeats the process.

The code also includes some commented-out lines that show how to put the Waspmote platform to deep sleep mode for a certain period of time. This feature can be used to save power when the sensor readings are not needed for long periods.

Overall, this code is an example of how to read sensor values using the Waspmote platform and print them to the USB port.

Graphical user interface, text, application

Description automatically generated

17: SMART\_CITIES\_TX\_ALL:

#include <WaspSensorCities\_PRO.h>

#include <WaspXBee802.h>

#include <WaspFrame.h>

/\* https://github.com/NawafAlzahem/Capstone-Project.git \*/

// TODO: Only broadcast seems to work

char RX\_ADDRESS[] = "000000000000FFFF"; /\* BROADCAST \*/

//char RX\_ADDRESS[] = "0013A20041678A0E"; /\* Robby1 MEshlium MAC \*/

//char RX\_ADDRESS[] = "0013A2004149DA14"; //XBEE S1 from IoT lab

//char RX\_ADDRESS[] = "0013A2004149DA23";

/\*

\* NOTE Nov 08, 2022:

\* Only NET address seems to work.

\* MAC Address works only for broadcast,

\* not for specific address

\*/

char MY\_NET\_ADDRESS[]="5220";

//char RX\_NET\_ADDRESS[]="1221";

char RX\_NET\_ADDRESS[]="1111"; // MESHLIUM NET ADDRESS

/\* In 802.15.4 you can use either full 8 HEX standard MAC

\* Address, or you can use the 2 HEX NET address.

\* The receiver will adapt automatically.

\* 0= use NET ADDRESS, 1 = USE MAC \*/

uint8\_t ADDRESS\_TYPE = 1;

char NODE\_ID[] = "TX\_SMART\_CITIES";

uint8\_t PANID[2] ; /\* Set the PAN\_ID global variable \*/

char node\_data[20];

/\* MAKE SURE YOU FIND THE CORRECT SOCKET! \*/

bmeCitiesSensor bme(SOCKET\_E);

float temp; // temperature in ºC

float hum; // realitve humidity in %RH

float pres; // pressure in Pa

uint8\_t error;

int counter = 0;

int delayTime;

uint8\_t api\_mode = 1; /\* 0-2 \*/

uint8\_t mac\_mode = 0; /\* 0-3 \*/

void setup(){

USB.ON();

USB.println(F("\n\*\*\*\*\*\*\* 802.15.4 SEND (TX) FULL \*\*\*\*\*\*\n"));

xbee802.ON();

PANID[0]=0x33;

PANID[1]=0x32;

/\* Only if you need to set the ID \*/

if(xbee802.setPAN(PANID)!=0){USB.println(F("ERROR setting PAN\_ID..."));}

error=xbee802.getPAN();

if(error!=0){ /\* Only if it didn't get the ID \*/

USB.println(F("ERROR getting PAN\_ID..."));

}

USB.print(F("---> PAN\_ID: "));

for(int i=0;i<2;i++){

USB.printHex(xbee802.PAN\_ID[i]);

}

USB.println();

xbee802.setChannel(0xC);

xbee802.getChannel();

USB.print(F("---> xbee802.channel: "));

USB.printHex(xbee802.channel); /\* IT SHOULD BE CHANNEL C \*/

USB.println();

xbee802.setNodeIdentifier(NODE\_ID);

xbee802.getNodeIdentifier();

USB.print(F("---> node ID: "));

USB.println(xbee802.nodeID); /\* Should print the whole array \*/

RTC.ON(); /\* Mote's time \*/

/\* TESTED, WORKS FINE! \*/

error = xbee802.setRTCfromMeshlium("0013A20041678A0E"); //Robby1

if( error != 0 ){ /\* Only in case of error...\*/

USB.println(F("SET RTC Time from Meshlium error..."));

}

USB.print(F("---> API mode: "));

USB.println(api\_mode, DEC);

/\* Set API mode accordingly \*/

xbee802.setAPI(api\_mode); /\* WHEN SET OTHERWISE, IT IS NOT WORKING !! \*/

/\* to select at any time if the modules are to use

\* a totally compatible heading format \*/

xbee802.setMacMode(mac\_mode); /\* 0-3 \*/

USB.print(F("---> MAC mode: "));

USB.println(xbee802.getMacMode(), DEC);

USB.print("---> My MAC ADDRESS: ");

xbee802.getOwnMacHigh();

xbee802.getOwnMacLow();

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacHigh[i]);

}

for (int i=0;i<4;i++){

USB.printHex(xbee802.sourceMacLow[i]);

}

USB.println();

USB.print(F("---> setOwnNetAddress("));

USB.print(MY\_NET\_ADDRESS);

if(xbee802.setOwnNetAddress(MY\_NET\_ADDRESS) == 0){

USB.print(F(") SUCCESSFUL..."));

} else{

USB.print(F(") ERROR!..."));

}

USB.println();

// Show '\_serial\_id' stored by the API when powering up

USB.print(F("---> Serial ID: "));

for (int i=0;i<8;i++){

USB.printHex(\_serial\_id[i]);

}

USB.println();

USB.print(F("---> RTC Time:"));

USB.println(RTC.getTime());

}

void loop(){

USB.println(F("\n\*\*\*\*\*\*\*\*\*\*\*\*\*\* LOOP STARTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

bme.ON();

temp = bme.getTemperature();

hum = bme.getHumidity();

pres = bme.getPressure();

USB.println(F("\*\*\*\*\*\*\*\*\*\*\* TEMP-HUM-PRESS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"));

USB.print(F("Temperature: "));

USB.printFloat(temp, 2);

USB.println(F(" C"));

USB.print(F("Relative Humidity: "));

USB.printFloat(hum, 2);

USB.println(F(" %"));

USB.print(F("Pressure: "));

USB.printFloat(pres, 2);

USB.println(F(" Pa"));

USB.println(F("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"));

counter++;

sprintf(node\_data,"Packet No:%d",counter);

USB.println(F("-------- 1. Create ASCII frame ----------"));

frame.createFrame(ASCII, NODE\_ID); /\* Set the Node ID for the frame \*/

frame.setFrameSize(125);

/\* EXAMPLE\_FRAME

TIMEOUT\_FRAME

EVENT\_FRAME

ALARM\_FRAME

SERVICE1\_FRAME

SERVICE2\_FRAME \*/

frame.setFrameType(EVENT\_FRAME);

frame.addSensor(SENSOR\_STR, node\_data);

frame.addSensor(SENSOR\_IN\_TEMP, temp);

frame.addSensor(SENSOR\_CITIES\_PRO\_HUM, hum);

frame.addSensor(SENSOR\_CITIES\_PRO\_PRES, pres);

frame.addSensor(SENSOR\_BAT, PWR.getBatteryLevel());

USB.println(F("---> Frame 2 send: ")); frame.showFrame();

if(ADDRESS\_TYPE == 0){ /\* 0 or 1 \*/

USB.print(F("Sending to NET address: "));

USB.println(RX\_NET\_ADDRESS);

error = xbee802.send(RX\_NET\_ADDRESS, frame.buffer, frame.length);

}else{

USB.print(F("Sending to MAC address: "));

USB.println(RX\_ADDRESS);

error = xbee802.send(RX\_ADDRESS, frame.buffer, frame.length);

}

if( error == 0 ){ // check TX flag

USB.println(F("---> frame sent ok"));

Utils.blinkGreenLED();

/\* it should deteriorate with distance \*/

USB.print(F("---> RSSI(dBm): "));

USB.println(xbee802.\_rssi);

/\* Receive XBee packet (wait message for 2 seconds) \*/

error = xbee802.receivePacketTimeout( 4000 );

if( error == 0 ){

USB.println(F("------- 1.1 Responce packet received ---------"));

/\* define the variable ad hoc \*/

char data\_in[xbee802.\_length]; // assign the length correct value

/\* Show data stored in '\_payload' buffer indicated by '\_length' \*/

USB.print(F("---> Received message: "));

for(int i=0;i<xbee802.\_length;i++){

USB.print(xbee802.\_payload[i]);

}

USB.println();

}

} else {

USB.println(F("---> OOPS..., frame sent error"));

Utils.blinkRedLED();

}

bme.OFF(); /\* Obviously saves battery? \*/

delayTime = 5; /\* seconds \*/

USB.print(F("\n\*\*\*\*\* LOOP END, Wait for "));

USB.print(delayTime);

USB.print(F(" sec \*\*\*\*\*\*\*\*\*\n"));

delay(delayTime \* 1000); /\* milliseconds \*/

}

Detailed explanation of the functionality of the code:

This code is for the Waspmote device, which enables wireless communication using the XBee 802.15.4 modules. The code sends data from various sensors (including temperature, humidity, pressure, and battery level) to a specific address (either a MAC address or a network address) using a frame of ASCII characters. The setup function initializes the communication settings, including setting the PAN ID and node ID, and sets the address type. The loop function retrieves data from the sensors and creates a frame to send the data. It then sends the frame to the specified address and receives a response packet if there is one. It waits for a specified amount of time before looping again.

Graphical user interface, text

Description automatically generated with medium confidence

1. github link and barcode

link: <https://github.com/NawafAlzahem/Capstone-Project.git>

barcode:

Qr code

Description automatically generated

Next step now is developing the android studio application development for usage of biometrics :

A screenshot of a computer

Description automatically generated with medium confidence

