Annex A

El present Annex conté tots els codis finals relatius a la primera part pràctica del projecte: *Aplicacions de l'ESP32 i ESP8266 en mode STA*.

A1. Connexió de l'ESP8266 i ESP32 a una xarxa WiFi en mode STA

A continuació es mostren els codis que permeten configurar tant el mòdul ESP32 com l'ESP8266 en mode STA.

ESP32 - STA.ino

```
// Definim les llibreries necessàries

#include <SPI.h>
#include <WiFiNINA.h>

// Establim les credencials de la nostra xarxa local

const char* ssid     = "MiFibra-ACDB";
const char* password = "mYnsjgb5";

// Inicialitzem el port sèrie i el mòdul WiFi de l'ESP32 en mode STA

void setup()
{
  Serial.begin(115200);
  delay(10);
  WiFi.begin(ssid, password);
  Serial.print("Connectant a:\t");
  Serial.println(ssid);

  // Esperem a estar connectats

  while (WiFi.status() != WL_CONNECTED)
  {
    delay(200);
    Serial.print('.');
  }

  //  Mostrem missatge d'èxit conforme ens hem connectat a la xarxa
  //  També mostrem l'adreça IP d'aquesta

  Serial.println();
  Serial.print("Connectat a:\t");
  Serial.println(WiFi.SSID());
  Serial.print("Adreça IP:\t");
  Serial.println(WiFi.localIP());
}

void loop()
{}
```

ESP8266 - STA.ino

```
// Definim les llibreries necessàries

#include <ESP8266WiFi.h>

// Establim les credencials de la nostra xarxa local

const char* ssid     = "MiFibra-ACDB";
const char* password = "mYnsjgb5";

// Inicialitzem el port sèrie i el mòdul WiFi de l'ESP8266 en mode AP

void setup()
{
  Serial.begin(115200);
  delay(10);
  WiFi.begin(ssid, password);
  Serial.print("Connectant a:\t");
  Serial.println(ssid);

  // Esperem a estar connectats

  WiFi.mode(WIFI_STA);
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(200);
    Serial.print('.');
  }

  //  Mostrem missatge d'èxit conforme ens hem connectat a la xarxa
  //  També mostrem l'adreça IP d'aquesta

  Serial.println();
  Serial.print("Connectat a:\t");
  Serial.println(WiFi.SSID());
  Serial.print("Adreça IP:\t");
  Serial.println(WiFi.localIP());
}

void loop()
{
}
```

A2. Primera aplicació: Controlar estat LED BUILTIN (ESP8266 i ESP32)

A continuació es mostra el codi que ha permès dur a terme la primera aplicació en mode STA.

LED_BUILTIN-ESP32 - STA.ino

```
// Definim les llibreries necessàries

#include <SPI.h>
#include <WiFiNINA.h>
```

```
// Llibreria definida per nosaltres i que permet emmagatzemar l'SSID i
contrasenya de la xarxa local, per garantir una mica més de seguretat

#include "SECRET.h"

char ssid[] = SECRET_SSID;
char password[] = SECRET_PASS;

// Obrim el port 80 (port per a transmissions HTTP) del nostre
encaminador, perquè el servidor es pugui comunicar amb els clients

WiFiServer server(80);

// Inicialitzem el port sèrie

void setup() {
  Serial.begin(115200);

  // Prepararem el LED

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, 0);

  // Ens connectem a la xarxa WiFi

  Serial.println();
  Serial.println();
  Serial.print(F("Connectant a "));
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }
  Serial.println();
  Serial.println(F("Connexió amb la xarxa establerta"));

  // Iniciem el servidor

  server.begin();
  Serial.println(F("Servidor creat"));

  // Mostrem pel port sèrie l'adreça IP

  Serial.println(WiFi.localIP());
  Serial.print(F("Per modificar l'estat del LED consulta l'adreça
http://"));
  Serial.println(WiFi.localIP());
}

void loop() {

  // Comprovem si s'ha connectat cap client

  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  Serial.println(F("Client nou"));
  client.setTimeout(5000);
```

```
  // Llegim la primera línia de la sol·licitud

  String req = client.readStringUntil('\r');
  Serial.println(F("Sol·licitud: "));
  Serial.println(req);

  // En base a la sol·licitud rebuda, modifiquem l'estat de la
variable "val"

  int val;
  if (req.indexOf(F("/builtin/1")) != -1) {
    val = 1;
  } else if (req.indexOf(F("/builtin/0")) != -1) {
    val = 0;
  } else {
    val = digitalRead(LED_BUILTIN);
  }

  // Establim l'estat del LED segons la sol·licitud

  digitalWrite(LED_BUILTIN, val);

  // Mentre el client es trobi disponible segueix llegint peticions
rebudes del servidor

  while (client.available()) {
    client.read();
  }

  // En base a les peticions rebudes, el client actualitza la web

  client.print(F("HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nEL LED BUILTIN
ACTUALMENT ESTA "));
  client.print((val) ? F("ENCES") : F("APAGAT"));
  client.print(F("<br><br>APRETA <a href='http://"));
  client.print(WiFi.localIP());
  client.print(F("/builtin/1'>AQUI</a> PER ENCENDRE EL LED BUILTIN, O
<a href='http://"));
  client.print(WiFi.localIP());
  client.print(F("/builtin/0'>AQUI</a> PER APAGAR EL LED
BUILTIN.</html>"));

  // Un cop realitzada l'acció encomanada, el servidor es desconnecta
del client a l'espera de rebre noves peticions

  Serial.println(F("Desconnectant del client"));

}
```

LED_BUILTIN-ESP8266 - STA.ino

```
// Definim les llibreries necessàries

#include <ESP8266WiFi.h>

// Llibreria definida per nosaltres i que permet emmagatzemar l'SSID i
contrasenya de la xarxa local, per garantir una mica més de seguretat

#include "SECRET.h"

char ssid[] = SECRET_SSID;
```

```
char password[] = SECRET_PASS;

// Obrim el port 80 (port per a transmissions HTTP) del nostre
encaminador, perquè el servidor es pugui comunicar amb els clients

WiFiServer server(80);


// Inicialitzem el port sèrie

void setup() {
  Serial.begin(115200);

  // Preparem el LED

  pinMode(LED_BUILTIN, OUTPUT);
  digitalWrite(LED_BUILTIN, 0);

  // Ens connectem a la xarxa WiFi

  Serial.println();
  Serial.println();
  Serial.print(F("Connectant a "));
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  WiFi.mode(WIFI_STA);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(F("."));
  }
  Serial.println();
  Serial.println(F("Connexió amb la xarxa establerta"));

  // Iniciem el servidor

  server.begin();
  Serial.println(F("Servidor creat"));

 // Mostrem pel port sèrie l'adreça IP

  Serial.println(WiFi.localIP());
  Serial.print(F("Per modificar l'estat del LED consulta l'adreça
http://"));
  Serial.println(WiFi.localIP());
}

void loop() {

  // Comprovem si s'ha connectat cap client

  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  Serial.println(F("Client nou"));
  client.setTimeout(5000);

  // Llegim la primera línia de la sol·licitud

  String req = client.readStringUntil('\r');
  Serial.println(F("Sol·licitud: "));
```

```
  Serial.println(req);

  // En base a la sol·licitud rebuda, modifiquem l'estat de la
variable "val"

  int val;
  if (req.indexOf(F("/builtin/1")) != -1) {
    val = 0;
  } else if (req.indexOf(F("/builtin/0")) != -1) {
    val = 1;
  } else {
    val = digitalRead(LED_BUILTIN);
  }

  // Establim l'estat del LED segons la sol·licitud

  digitalWrite(LED_BUILTIN, val);

  // Mentre el client es trobi disponible segueix llegint peticions
rebudes del servidor

  while (client.available()) {
    client.read();
  }

  // En base a les peticions rebudes, el client actualitza la web

  client.print(F("HTTP/1.1 200 OK\r\nContent-Type:
text/html\r\n\r\n<!DOCTYPE HTML>\r\n<html>\r\nEL LED BUILTIN
ACTUALMENT ESTA "));
  client.print((val) ? F("APAGAT") : F("ENCES"));
  client.print(F("<br><br>APRETA <a href='http://"));
  client.print(WiFi.localIP());
  client.print(F("/builtin/1'>AQUI</a> PER ENCENDRE EL LED BUILTIN, O
<a href='http://"));
  client.print(WiFi.localIP());
  client.print(F("/builtin/0'>AQUI</a> PER APAGAR EL LED
BUILTIN.</html>"));

  // Un cop realitzada l'acció encomanada, el servidor es desconnecta
del client a l'espera de rebre noves peticions

  Serial.println(F("Desconnectant del client"));
}
```

A3. Segona aplicació: Visualitzar la temperatura i humitat mitjançant el sensor DHT11 i un servidor web (ESP8266 i ESP32)

A continuació es mostra el codi que ha permès dur a terme la segona aplicació en mode STA.

TEMPERATURA - STA.ino

```
// Definim les llibreries necessàries

#include <ESP8266WiFi.h>
#include <ESP8266WebServer.h>
```

```
#include "DHT.h"

// Llibreria que ens permetrà fer servir funcions amb què controlar el
sensor DHT11

#define DHTTYPE DHT11

// Definim l'SSID i la contrasenya de la xarxa local

const char* ssid = "MiFibra-ACDB";
const char* password = "mYnsjgb5";

// Obrim el port 80 (port per a transmissions HTTP) del nostre
encaminador, perquè el servidor es pugui comunicar amb els clients

ESP8266WebServer server(80);

// Definim en quin pin digital connectarem el sensor DHT11

uint8_t DHTPin = D4;

// Configurem el sensor DHT11

DHT dht(DHTPin, DHTTYPE);

// Definim les variables que contindran els valors de temperatura i
humitat

float Temperatura;
float Humitat;
float maxTemperatura;
float minTemperatura;
float maxHumitat;
float minHumitat;

// Inicialitzem el port sèrie

void setup() {
  Serial.begin(115200);
  delay(100);

  // Definim el sensor DHT11 com a entrada

  pinMode(DHTPin, INPUT);

  // Inicialitzem el sensor DHT11

  dht.begin();

  // Ens connectem a la xarxa WiFi

  Serial.println("Connectant a ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  WiFi.mode(WIFI_STA);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connectat!");
```

```
  Serial.print("Adreça IP: ");  Serial.println(WiFi.localIP());

  // Indiquem que, quan un servidor rebi una petició HTTP en la
ruta  arrel(/),  activarà la funció handle_OnConnect, la qual es
genera quan la connexió del client amb el servidor s'estableix.
Aquesta funció podrà llegir o escriure per intercanviar informació amb
el client

  server.on("/", handle_OnConnect);

  // Si el client sol·licita qualsevol URL diferent de l' especificat,
el servidor respondrà amb un estat HTTP 404 (No trobat) i un missatge
d'error per a l'usuari

  server.onNotFound(handle_NotFound);

  // Inicialitzem el servidor

  server.begin();
  Serial.println("Servidor HTTP iniciat");
  setupVariables();

}
void loop() {

  // Per controlar les peticions HTTP entrants, necessitem aquesta
funció

  server.handleClient();

  // Llegim la temperatura i humitat

  float Temperatura = dht.readTemperature();
  float Humitat = dht.readHumidity();
  temphumMinMax(Temperatura, Humitat);
}
void setupVariables() {
  maxTemperatura = 0.0;
  minTemperatura = 0.0;
  maxHumitat = 0.0;
  minHumitat = 0.0;
}

// Obtenim el valor de la temperatura i humitat màxima i mínima

void temphumMinMax(float Temperatura, float Humitat) {
  if (maxTemperatura < Temperatura) {
    maxTemperatura = Temperatura;
  }

  if (minTemperatura == 0.0) {
    minTemperatura = Temperatura;
  }

  if (minTemperatura > Temperatura) {
    minTemperatura = Temperatura;
  }

  if (maxHumitat < Humitat) {
    maxHumitat = Humitat;
  }
```

```
  if (minHumitat == 0.0) {
    minHumitat = Humitat;
  }

  if (minHumitat > Humitat) {
    minHumitat = Humitat;
  }
}

// Creem la funció que havíem definit anteriorment, la qual es genera
quan la connexió del client amb el servidor s'estableix

void handle_OnConnect() {
  Temperatura = dht.readTemperature();
  Humitat = dht.readHumidity();

  // Per respondre a la petició HTTP, fem servir el mètode "send". En
aquest cas, enviem el codi 200 (equivalent a la resposta OK), després
especifiquem el tipus de contingut com "text/html" i, per últim,
mitjançant la funció SendHTML(), creem una pàgina HTML dinàmica que
contindrà tots els valors de temperatura i humitat

  server.send(200, "text/html", SendHTML(Temperatura, Humitat,
maxTemperatura, minTemperatura, maxHumitat, minHumitat ));
}

// Creem la funció que havíem definit anteriorment, la qual retorna un
missatge d'error

void handle_NotFound() {
  server.send(404, "text/plain", "Not found");
}

// Generem una pàgina web cada cop que el servidor web del nostre
microcontrolador rep una petició del client. La funció agafa els
valors de la temperatura i humitat (amb els valors màxims i mínims) i
genera dinàmicament el contingut HTML

String SendHTML(float Temperatura, float Humitat, float
maxTemperatura, float minTemperatura, float maxHumitat, float
minHumitat) {
  String ptr = "<!DOCTYPE html> <html>\n";
  ptr += "<head><meta name=\"viewport\" content=\"width=device-width,
initial-scale=1.0, user-scalable=no\">\n";
  ptr += "<title>Informe del temps (TFG)</title>\n";
  ptr += "<meta http-equiv='refresh' content='5'>";
  ptr += "<style>html { font-family: cursive; display: inline-block;
margin: 0px auto; text-align: center;}\n";
  ptr += "body{margin-top: 50px;} h1 {font-size: 40px; color:
#444444;margin: 100px auto 60px;}\n";
  ptr += "p {font-size: 30px;color: #444444;margin-bottom: 10px;}\n";
  ptr += "f {font-size: 24px;color: #06afcb;margin-bottom: 20px;}\n";
  ptr += "c {font-size: 24px;color: #fb0606;margin-bottom: 20px;}\n";
  ptr += "n {font-size: 24px;color: #131212;margin-bottom: 20px;}\n";
  ptr += "</style>\n";
  ptr += "</head>\n";
  ptr += "<body>\n";
  ptr += "<div id=\"Contingut\">\n";
  ptr += "<h1>Informe del temps</h1>\n";
  ptr += "<p>Temperatura actual: ";
  ptr += (float)Temperatura;
```

```
  ptr += "&#186C</p>";
  ptr += "<c>Temperatura m&#224xima: ";
  ptr += (float)maxTemperatura;
  ptr += "&#186C </c>";
  ptr += "<f>Temperatura m&#237nima: ";
  ptr += (float)minTemperatura;
  ptr += "&#186C</f>";
  ptr += "<p>Humitat actual: ";
  ptr += (float)Humitat;
  ptr += "%</p>";
  ptr += "<c>Humitat m&#224xima: ";
  ptr += (float)maxHumitat;
  ptr += "% </c>";
  ptr += "<f>Humitat m&#237nima: ";
  ptr += (float)minHumitat;
  ptr += "%</f>";
  ptr += "</div>\n";
  ptr += "</body>\n";
  ptr += "</html>\n";
  return ptr;
}
```

Annex B

El present Annex conté tots els codis finals relatius a la segona part pràctica del projecte: *Aplicacions de l'ESP32 i ESP8266 en mode SoftAP*.

B1. Connexió de l'ESP8266 i ESP32 a una xarxa WiFi en mode AP

A continuació es mostren els codis que permeten configurar tant el mòdul ESP32 com l'ESP8266 en mode SoftAP.

ESP32 - SOFTAP.ino

```
// Definim les llibreries necessàries

#include <SPI.h>
#include <WiFiNINA.h>

// Definim l'SSID i la contrasenya que volem que tingui la nostra
xarxa

const char *ssid = "TFG - ESP32";
const char *password = "password";

void setup() {

  // Inicialitzem el port sèrie

  Serial.begin(115200);
  delay(10);
```

```
  // Iniciem la creació de la nostra xarxa

  while (!WiFi.beginAP(ssid, password))
  {
    Serial.println(".");
    delay(100);
  }

  // Creació satisfactòria. Mostrem per port sèrie l'SSID de la nostra
xarxa i la seva adreça IP

  IPAddress ip = WiFi.localIP();
  Serial.println("Mode softAP actiu ");
  Serial.print("Nom de la xarxa: ");
  Serial.println(ssid);
  Serial.print("Adreça IP: ");
  Serial.println(ip);

}

void loop() { }
```

ESP8266 - SOFTAP.ino

```
// Definim la llibreria necessària

#include <ESP8266WiFi.h>

// Definim l'SSID i la contrasenya que volem que tingui la nostra
xarxa

const char *ssid = "TFG - ESP8266";
const char *password = "password";

void setup() {

  // Inicialitzem el port sèrie

  Serial.begin(115200);
  delay(10);

  // Iniciem la creació de la nostra xarxa

  WiFi.mode(WIFI_AP);
  while (!WiFi.softAP(ssid, password))
  {
    Serial.println(".");
    delay(100);
  }

  // Creació satisfactòria. Mostrem per port sèrie l'SSID de la nostra
xarxa i la seva adreça IP

  Serial.println("Mode softAP actiu ");
  Serial.print("Nom de la xarxa: ");
  Serial.println(ssid);
  Serial.print("Adreça IP: ");
  Serial.println(WiFi.softAPIP());
}
```

```
void loop() { }
```

B2. Primera aplicació: Alarma amb pantalla LCD, teclat i WiFi (ESP32)

A continuació es mostra el codi que ha permès dur a terme la primera aplicació en mode SoftAP.

ALARMA - SOFTAP.ino

```
// Definim les llibreries necessàries

#include <WiFiNINA.h>

// Llibreria definida per nosaltres i que permet emmagatzemar l'SSID i
contrasenya de la xarxa local, per garantir una mica més de seguretat

#include "arduino_secrets.h"

// Llibreria que ens permetrà controlar la pantalla de cristall líquid
LCD 16x2

#include <LiquidCrystal.h>

// Llibreria que ens permetrà controlar el teclat matricial 4x4

#include <Keypad.h>

// Definim on connectarem físicament el brunzidor, el pin Trig i el
pin Echo. Aquests darrers pins són els que farem servir per controlar
el sensor d'ultrasons, on el Trig estarà connectat al pin 1 que
enviarà el pols ultrasònic, i l'Echo estarà connectat al pin 2, el
qual s'encarregarà de rebre el rebot del pols

#define altaveu 5
#define trigPin 1
#define echoPin 2

// Variable que farem servir per calcular la distància del primer
obstacle amb què reboti el pols. En base al temps que trigui en
rebotar el pols, podrem saber la distància. Long fa referència a que
la variable contindrà valors entre -2147483648 fins 2147483647

long duracio;

// La primera variable emmagatzemarà el resultat del càlcul de la
distància; la  segona emmagatzemarà la distància inicial a la qual es
col·loca el sensor; la tercera emmagatzemarà la distància actual
(gràcies a aquesta variable si passem una mà ho sabrà, ja que la
distància actual serà diferent de la inicial). Per acabar, la variable
i ens permetrà col·locar el cursor a la pantalla LCD

int distancia, distanciaInicial, distanciaActual, i;

// Variable que contindrà la configuració de com veurem el menú
principal
```

```
int menu = 0;

// Variable que indicarà si l'alarma esta activada o no pel servidor
web

int estat;

// Definim quina serà la contrasenya pre-establerta de la nostra
alarma

String contrasenya = "1234";

//  Definim una variable que contindrà la contrasenya introduïda per
l'usuari

String contasenyaIntroduida;

// Determina l'estat de l'alarma (activada o desactivada). Aquesta
variable és de tipus booleà, per la qual cosa tindrà dos valors: fals
o vertader

boolean activada = false;

// Com a variable booleana que és, quan aquesta té valor vertader,
activa l'alarma

boolean activarAlarma = false;

// Aquesta variable, quan té valor vertader, activa la funció que
s'encarrega de mesurar la distància actual. Per tant, aquesta només
funcionarà quan l'alarma estigui activada

boolean alarmaActivada = false;

// Variable gràcies a la qual podrem entrar en el mode de canvi de
contrasenya

boolean contrasenyaCanviada = false;

// Variable gràcies a la qual podrem efectuar el canvi de contrasenya

boolean modeCanviContrasenya = false;

// Definim el nombre de files i columnes que té el nostre teclat
matricial. Les definim com a const byte, ja que la variable mai pot
canviar de valor

const byte FILES = 4; // 4 files
const byte COLUMNES = 4; // 4 columnes

// Variable que detectarà les tecles que premem en el teclat

char teclaPremuda;

// Definim les tecles que té el nostre teclat, separades per files i
columnes

char keyMap[FILES][COLUMNES] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
```

```
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

// Definim on connectarem físicament els pins pertanyents a les files
i columnes del teclat matricial

byte pinsFila[FILES] = {15, 16, 17, 18};
byte pinsColumna[COLUMNES] = {19, 20, 21, 0};

// Definim el nostre teclat amb totes les variables necessàries

Keypad myKeypad = Keypad( makeKeymap(keyMap), pinsFila, pinsColumna,
FILES, COLUMNES);

// Definim on connectarem físicament els pins que permeten fer
funcionar la pantalla LCD

LiquidCrystal lcd(3, 4, 8, 9, 10, 11); // Paràmetres: (rs, enable, D4,
D5, D6, D7)

// Establim el nom SSID i la contrasenya que li volem donar a la
xarxa. Per seguretat, aquesta informació es troba en una pestanya a
part, però vinculem l'SSID i la contrasenya al programa principal fent
servir les variables SECRET_SSID i SECRET_PASS

char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;

// Estat temporal de la xarxa quan es crida WiFi.begin() i es manté
actiu fins que caduca el nombre d'intents (resultant en
WL_CONNECT_FAILED) o fins que s'estableixi una connexió (resultant
WL_CONNECTED)

int status = WL_IDLE_STATUS;

// Creació d'un servidor en el port 80

WiFiServer server(80);

void setup() {

  // Iniciem la pantalla LCD amb setze columnes i dues files

  lcd.begin(16, 2);

  // Definim com a entrades o sortides els dispositius

  pinMode(altaveu, OUTPUT); /
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);

  // Connectem un LED verd al pin 6 i un vermell al pin 7

  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  delay(10);

  // Inicialitzem port sèrie

  Serial.begin(9600);
```

```
  // Espera que obrim el port sèrie per inicialitzar el programa

  while (!Serial) {
    ;
  }
  Serial.println("Mode softAP actiu");

  // Comprovem l'estat del mòdul WiFi

  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("El mode softAP no s'ha pogut activar!");

    // Mentre l'estat del WiFi sigui aquest, el port sèrie mostrarà el
missatge "El mode AP no s'ha pogut activar!" de forma reiterada

    while (true);
  }
  Serial.print("Creant xarxa en mode softAP amb nom: ");
  Serial.println(ssid);
  status = WiFi.beginAP(ssid, pass);
  if (status != WL_AP_LISTENING) {
    Serial.println("No s'ha pogut crear una xarxa en mode AP");

    // Si no s'ha pogut crear la xarxa, el programa no continua

    while (true);
  }

  // Esperem 10 segons per a la connexió

  delay(10000);

  // Iniciem el servidor web en el port 80

  server.begin();

  // Per poder veure pel port sèrie l'estat de la connexió, executem
aquesta funció

  estatWiFi();
}

void loop() {

  // Comparem l'estat anterior del WiFi amb l'estat actual

  if (status != WiFi.status()) {

    // Si canvia, actualitzem la variable "status"

    status = WiFi.status();

    // Detectem si algun dispositiu es connecta a la xarxa

    if (status == WL_AP_CONNECTED) {
      Serial.println("Dispositiu connectat a la xarxa");

      // En cas contrari, voldrà dir que el dispositiu s'haurà
desconnectat de la xarxa
```

```
    } else {
      Serial.println("Dispositiu desconnectat de la xarxa");
    }
  }

  // Definim la variable client, la qual indicarà si el servidor web
es troba disponible

  WiFiClient client = server.available();
  if (client) {                             // Si aconseguim un
client,
    String liniaActual = "";                // fem una cadena per
emmagatzemar les dades entrants del client
    while (client.connected()) {            // Bucle mentre el client
està connectat
      if (client.available()) {             // Si hi ha bytes a llegir
del client,
        char c = client.read();             // llegim un byte,
aleshores
        if (c == '\n') {                    //  si el byte és un
caràcter de línia nova, i la línia nova està en blanc, voldrà dir que
es el final de la petició HTTP del client. Per tant, enviarem la
resposta que veurem a continuació

          // Aquest és el final de la sol·licitud HTTP del client. Per
tant ,a continuació enviem la resposta en base a la seva petició

          if (liniaActual.length() == 0) {

            // Indiquem que estem enviant codi HTML

            client.println(F("<!DOCTYPE HTML>"));
            client.println(F("<html>"));
            client.println(F("<head>"));
            client.println(F("<title>ALARMA (TFG)</title>"));
            client.println(F("</head>"));
            client.println(F("<body> <h1> ALARMA AMB WIFI </h1>"));
            client.print(F("L&#39ALARMA EST&#192 "));
            client.print(F((estat) ? F("ACTIVADA") :

F("DESACTIVADA")));
            client.println(F("<p>Prem <a href=\"/H\">aqu&#237</a> per
encendre l&#39alarma<p>"));
            client.println(F("<p>Prem <a href=\"/L\">aqu&#237</a> per
apagar l&#39alarma<p>"));
            client.println();

            // Fem servir aquesta funció per poder sortir de
l'execució normal d'un bucle. D'aquesta manera aconseguim que cada cop
que activem l'alarma o no, el programa funcioni i no es quedi encallat
en el bucle

            break;
          }
          // Si obtenim una nova línia del client, aleshores esborrem
el contingut de la cadena líniaActual
          else {
            liniaActual = "";
          }
        }
```

```
        // Si obtenim alguna cosa més que un caràcter de retorn,
l'afegim al final de liniaActual

        else if (c != '\r') {
          liniaActual += c;
        }

        // Si liniaActual acaba en /H, aleshores activem l'alarma, i a
més a més mostrem al servidor web l'estat actual de l'alarma, és a
dir, ALARMA ACTIVADA. També mostrem per la pantalla LCD el menú
d'alarma activada

        if (liniaActual.endsWith("GET /H")) {
          activarAlarma = true;
          estat = 1;
        }

        // Si liniaActual acaba en /L, aleshores desactivem l'alarma,
i a més a més mostrem al servidor web l'estat actual de l'alarma, és a
dir, ALARMA DESACTIVADA. També mostrem per pantalla el menú inicial,
ja que representa que hem desactivat l'alarma

        if (liniaActual.endsWith("GET /L")) {
          alarmaActivada = false;
          activada = false;
          menu = 0;
          estat = 0;
        }
      }
    }

    // Aturem la connexió amb el client

    client.stop();
  }

  // Definim el comportament del programa quan activem l'alarma

  if (activarAlarma) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("L'alarma");
    lcd.setCursor(0, 1);
    lcd.print("s'activara en ");
    int compteEnrere = 9;
    while (compteEnrere != 0) {
      lcd.setCursor(14, 1);
      lcd.print(compteEnrere);
      compteEnrere--;

      // Indiquem el pin on tenim connectat l'altaveu, la freqüència
del seu so i la durada

      tone(altaveu, 700, 100);
      delay(1000);
    }
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Alarma activada!");
    distanciaInicial = obtenirDistancia();
    activarAlarma = false;
```

```
    alarmaActivada = true;
    digitalWrite(6, HIGH);
    digitalWrite(7, LOW);
  }

  // Definim què fa el programa quan l'alarma està sonant

  if (alarmaActivada == true) {
    distanciaActual = obtenirDistancia() + 10;
    if ( distanciaActual < distanciaInicial) {
      tone(altaveu, 1000);
      lcd.clear();
      introduirContrasenya();
    }
  }

  // Definim què fa el programa quan l'alarma no està activada

  if (alarmaActivada == false) {
    if (menu == 0 ) {
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print("A - ACTIVAR");
      lcd.setCursor(0, 1);
      lcd.print("B - CANVI CONTR.");
      menu = 1;
      digitalWrite(6, LOW);
      digitalWrite(7, LOW);
    }

    // Si premem la tecla A activem l'alarma

    teclaPremuda = myKeypad.getKey();
    if (teclaPremuda == 'A') {
      tone(altaveu, 1000, 200);
      activarAlarma = true;
    }

    // Si premem la tecla B canviem la contrasenya de  l'alarma

    else if (teclaPremuda == 'B') {
      lcd.clear();
      int i = 1; // Variable que ens permet col·locar el cursor a la
pantalla LCD
      tone(altaveu, 2000, 100);
      contasenyaIntroduida = "";
      lcd.setCursor(0, 0);
      lcd.print("Contrasenya act."); // Ens indica la pantalla LCD que
introduïm la   contrasenya actual
      lcd.setCursor(0, 1);
      lcd.print(">");
      modeCanviContrasenya = true;
      contrasenyaCanviada = true;
      while (contrasenyaCanviada) {
        teclaPremuda = myKeypad.getKey();
        if (teclaPremuda != NO_KEY) { // Detectem qualsevol tecla que
es premi (0-9)
          if (teclaPremuda == '0' || teclaPremuda == '1' ||
teclaPremuda == '2' || teclaPremuda == '3' ||
              teclaPremuda == '4' || teclaPremuda == '5' ||
teclaPremuda == '6' || teclaPremuda == '7' ||
```

```
              teclaPremuda == '8' || teclaPremuda == '9' ) {
            contasenyaIntroduida += teclaPremuda;
            lcd.setCursor(i, 1);
            lcd.print("*"); // Cada cop que es premi una tecla,
imprimirem un *
            i++;
            tone(altaveu, 2000, 100);
          }
        }
        if (i > 5 || teclaPremuda == '#') { // Si introduïm més de 4
dígits i/o premem la tecla #,netegem la pantalla

          contasenyaIntroduida = "";
          i = 1;
          lcd.clear();
          lcd.setCursor(0, 0);
          lcd.print("Contrasenya act.");
          lcd.setCursor(0, 1);
          lcd.print("> ");
        }
        if ( teclaPremuda == '*') { // Si premem * validem la nostra
contrasenya
          i = 1;
          tone(altaveu, 2000, 100);
          if (contrasenya == contasenyaIntroduida) {
            contasenyaIntroduida = "";
            lcd.clear();
            lcd.setCursor(0, 0);
            lcd.print("Contrasenya nova");
            lcd.setCursor(0, 1);
            lcd.print("> ");
            while (modeCanviContrasenya) {
              teclaPremuda = myKeypad.getKey();
              if (teclaPremuda != NO_KEY) { // Detectem qualsevol
tecla que es premi (0-9)
                if (teclaPremuda == '0' || teclaPremuda == '1' ||
teclaPremuda == '2' || teclaPremuda == '3' ||
                    teclaPremuda == '4' || teclaPremuda == '5' ||
teclaPremuda == '6' || teclaPremuda == '7' ||
                    teclaPremuda == '8' || teclaPremuda == '9' ) {
                  contasenyaIntroduida += teclaPremuda;
                  lcd.setCursor(i, 1);
                  lcd.print("*"); // Cada cop que es premi una tecla,
imprimirem un *
                  i++;
                  tone(altaveu, 2000, 100);
                }
              }
              if (i > 5 || teclaPremuda == '#') { // Si introduïm més
de 4 dígits i/o premem la tecla #, netegem la pantalla

                contasenyaIntroduida = "";
                i = 1;
                tone(altaveu, 2000, 100);
                lcd.clear();
                lcd.setCursor(0, 0);
                lcd.print("Contrasenya nova");
                lcd.setCursor(0, 1);
                lcd.print("> ");
              }
```

```
              if ( teclaPremuda == '*') { // Si premem * validem la
nostra contrasenya
                i = 1;
                tone(altaveu, 2000, 100);
                contrasenya = contasenyaIntroduida; // La variable
contrasenya pren per valor l'introduït per l'usuari

                modeCanviContrasenya = false;
                contrasenyaCanviada = false;
                menu = 0; // Tornem a la pantalla principal
              }
            }
          }
        }
      }
    }
  }
}

// Funció que executem quan l'alarma sona

void introduirContrasenya() {

  int k = 1;
  contasenyaIntroduida = "";
  activada = true;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(" *** ALARMA *** ");
  lcd.setCursor(0, 1);
  lcd.print(">");
  digitalWrite(6, LOW);
  digitalWrite(7, HIGH); // El LED vermell estarà encès mentre
l'alarma estigui sonant
  while (activada) {
    teclaPremuda = myKeypad.getKey();
    if (teclaPremuda != NO_KEY) { // Detectem qualsevol tecla que es
premi (0-9)
      if (teclaPremuda == '0' || teclaPremuda == '1' || teclaPremuda
== '2' || teclaPremuda == '3' ||
          teclaPremuda == '4' || teclaPremuda == '5' || teclaPremuda
== '6' || teclaPremuda == '7' ||
          teclaPremuda == '8' || teclaPremuda == '9' ) {
        contasenyaIntroduida += teclaPremuda;
        lcd.setCursor(k, 1);
        lcd.print("*"); // Cada cop que es premi una tecla, imprimirem
un *
        k++;
      }
    }
    if (k > 5 || teclaPremuda == '#') { // Si introduïm més de 4
dígits i/o premem la tecla #, netegem la pantalla
      contasenyaIntroduida = "";
      k = 1;
      lcd.clear();
      lcd.setCursor(0, 0);
      lcd.print(" *** ALARMA *** ");
      lcd.setCursor(0, 1);
      lcd.print(">");
    }
```

```
    if ( teclaPremuda == '*') { // Si premem * validem la nostra
contrasenya
      if ( contasenyaIntroduida == contrasenya ) {
        activada = false;
        alarmaActivada = false;
        noTone(altaveu);
        menu = 0; // Tornem a la pantalla principal
      }
      else if (contasenyaIntroduida != contrasenya) {
        lcd.setCursor(0, 1);
        lcd.print("Error! Torna-hi");
        delay(2000);
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print(" *** ALARMA *** ");
        lcd.setCursor(0, 1);
        lcd.print(">");
      }
    }
  }
}

// Definim funció del sensor ultrasònic

long obtenirDistancia() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duracio = pulseIn(echoPin, HIGH);
  distancia = duracio * 0.034 / 2;
  return distancia;
}

// Quan la xarxa ha estat creada, mostrem pel port sèrie tota la
informació necessària

void estatWiFi() {
  Serial.print("SSID: ");
  Serial.println(WiFi.SSID());
  IPAddress ip = WiFi.localIP();
  Serial.print("Adreça IP: ");
  Serial.println(ip);

  // Indiquem on cal anar per controlar l'alarma amb WiFi

  Serial.print("Per poder controlar l'alarma amb WiFi, visita la web
http://");
  Serial.println(ip);
}
```

B3. Segona aplicació: Control LED RGB amb pantalla OLED (I2C) i WiFi (ESP8266)

A continuació es mostra el codi que ha permès dur a terme la segona aplicació en mode SoftAP.

CONTROL - RGB - SOFTAP.ino

```
// Definim les llibreries necessàries

#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>
#include <Wire.h>

// Gràcies a aquesta llibreria podrem controlar les funcions de la
pantalla OLED

#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Definim l'amplada i alçada de la pantalla, en píxels

#define SCREEN_WIDTH 128 // Amplada pantalla OLED en píxels
#define SCREEN_HEIGHT 64 // Alçada pantalla OLED en píxels

// Inicialitzem la pantalla amb tots els seus paràmetres

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// // Definim l'SSID i la contrasenya que volem que tingui la nostra
xarxa

const char *ssid = "TFG - ESP8266";
const char *password = "password";

// Obrim el port 80 (port per a transmissions HTTP), perquè el
servidor es pugui comunicar amb els clients

WiFiServer server(80);

// Definim variables del programa per fer-les servir més endavant

String redString = "0";
String greenString = "0";
String blueString = "0";
int pos1 = 0;
int pos2 = 0;
int pos3 = 0;
int pos4 = 0;

// Variable per emmagatzemar la sol·licitud HTTP

String header;

// Definim on connectarem els pins del LED RGB
```

```
const int R = D6;     // Pin vermell el connectarem al pin digital 6
const int G = D7;    // Pin verd el connectarem al pin digital 7
const int B = D8;     // Pin blau el connectarem al pin digital 8

// Variable que farem servir per fer una interacció de colors quan ens
creem la xarxa

int _step   =  1;

// Configuració de la resolució de bits PWM
const int resolution = 256;

// Temps actual

unsigned long currentTime = millis();

// Temps previ

unsigned long previousTime = 0;

// Definim el temps d'espera en mil·lisegons (exemple: 2000ms = 2s)

const long timeoutTime = 2000;

// Creem un parell de funcions que executaran una transició de colors
quan es creï la xarxa

int FadeOn(int pin)
{
  int fadeValue;
  for (fadeValue = 0 ; fadeValue <= 255; fadeValue += _step) {
    analogWrite(pin, fadeValue);

    // Esperem per veure l'efecte d'atenuació

    delay(6);
  }
  analogWrite(pin, 255);
  return (fadeValue);
}

int FadeOff(int pin)
{
  int fadeValue;
  // Ramp down pin
  for (fadeValue = 255 ; fadeValue >= 0; fadeValue -= _step) {
    analogWrite(pin, fadeValue);
    delay(6);
  }
  analogWrite(pin, 0);
  return (fadeValue);
}

void setup() {

  // Definim els pins del LED RGB com a sortides

  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);
```

```
  // Configurem la resolució del LED i configurem els pins en estat
baix

  analogWriteRange(resolution);
  analogWrite(R, 0);
  analogWrite(G, 0);
  analogWrite(B, 0);

  // Iniciem el mode SoftAP

  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);

  // Iniciem la pantalla OLED amb l'adreça 0x78 corresponent a la seva
direcció I2C

  display.begin(SSD1306_SWITCHCAPVCC, 0x78 >> 1);

  // Configurem les pantalles de visualització de l'SSD1306

  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  //display.setTextColor(BLACK, WHITE);
  display.setCursor(0, 0);
  display.println("  ESP8266");
  display.setTextSize(3);
  display.setTextColor(WHITE);
  display.setCursor(0, 20);
  display.println("LED RGB");
  display.setTextSize(1);
  display.setTextColor(WHITE);
  display.setCursor(0, 52);
  display.println("Treball de Fi de Grau");

  display.display();
  delay(5000);

  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  display.setCursor(0, 0);
  display.println("Connectant");
  display.clearDisplay();
  display.setTextSize(1);       display.setTextColor(WHITE);
  display.setCursor(0, 0);       display.print("SSID:");
display.println(ssid);
  display.setTextSize(2);       display.setTextColor(WHITE);
  display.setCursor(0, 18);      display.println(WiFi.softAPIP());
  display.display();

  // Inicialitzem el servidor

  server.begin();

  // Un cop finalitzada la creació de la xarxa, es procedeix a la
visualització de transicions de colors de l'RGB

  analogWrite(R, 0);
  analogWrite(G, 0);
  FadeOn(B);
```

```
  FadeOn(R);
  FadeOff(B);
  FadeOn(G);
  FadeOff(R);
  FadeOn(B);
  FadeOff(G);
  analogWrite(R, 255);
  analogWrite(G, 255);
  analogWrite(B, 255);
}
void loop() {

  // Definim la variable client, la qual indicarà si el servidor web
es troba disponible

  WiFiClient client = server.available();
  if (client) {                             // Si es connecta un nou
client,
    currentTime = millis();
    previousTime = currentTime;
    String currentLine = "";                // farem una cadena per
emmagatzemar les dades entrants del client
    while (client.connected() && currentTime - previousTime <=
timeoutTime) {            // Bucle mentre el client està connectat
      currentTime = millis();
      if (client.available()) {             // Si hi ha bytes a llegir
del client,
        char c = client.read();             // llegim un byte,
aleshores
        header += c;
        if (c == '\n') {                    //  si el byte és un
caràcter de línia nova, i la línia nova està en blanc, voldrà dir que
és el final de la petició HTTP del client. Per tant, enviarem la
resposta que veurem a continuació
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Indiquem que estem enviant codi HTML

            client.println("<!DOCTYPE html><html>");
            client.println("<head><meta name=\"viewport\"
content=\"width=device-width, initial-scale=1\">");
            client.println("<link rel=\"icon\" href=\"data:,\">");
            client.println("<link rel=\"stylesheet\"
href=\"https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstra
p.min.css\">");
            client.println("<script
src=\"https://cdnjs.cloudflare.com/ajax/libs/jscolor/2.0.4/jscolor.min
.js\"></script>");
            client.println("</head><body><div class=\"container\"><div
class=\"row\"><h1>ESP8266 RGB LED</h1></div>");
            client.println("<a class=\"btn btn-primary btn-lg\"
href=\"#\" id=\"change_color\" role=\"button\">Canvia Color</a> ");
            client.println("<input class=\"jscolor
{onFineChange:'update(this)'}\" id=\"rgb\"></div>");
            client.println("<script>function update(picker)
{document.getElementById('rgb').innerHTML = Math.round(picker.rgb[0])
```

```
+ ', ' +  Math.round(picker.rgb[1]) + ', ' +
Math.round(picker.rgb[2]);");
            client.println("document.getElementById(\"change_color\").
href=\"?r\" + Math.round(picker.rgb[0]) + \"g\"
+  Math.round(picker.rgb[1]) + \"b\" + Math.round(picker.rgb[2]) +
\"&\";}</script></body></html>");
            client.println();

            // Exemple de sol·licitud: /?r201g32b255&
            // Red = 201 | Green = 32 | Blue = 255

            if (header.indexOf("GET /?r") >= 0) {
              pos1 = header.indexOf('r');
              pos2 = header.indexOf('g');
              pos3 = header.indexOf('b');
              pos4 = header.indexOf('&');
              redString = header.substring(pos1 + 1, pos2);
              greenString = header.substring(pos2 + 1, pos3);
              blueString = header.substring(pos3 + 1, pos4);
              analogWrite(R, redString.toInt());
              analogWrite(G, greenString.toInt());
              analogWrite(B, blueString.toInt());

              // Mostrem per la pantalla OLED la informació de la
sol·licitud

              display.clearDisplay();
              display.setTextColor(WHITE);
              display.setTextSize(2);
              display.setTextColor(WHITE);
              display.setCursor(0, 0);
              display.print("R =
");  display.println(redString.toInt());
              display.setCursor(0, 18);
              display.print("G =
");  display.println(greenString.toInt());
              display.setCursor(0, 36);
              display.print("B =
");  display.println(blueString.toInt());
              display.display();

            }
            // Fem servir aquesta funció per poder sortir de
l'execució normal d'un bucle.
            break;
          } else { // si obtenim una nova sol·licitud, borrem l'actual
            currentLine = "";
          }
        } else if (c != '\r') {  // i si obtenim alguna cosa més que
un caràcter de retorn,
          currentLine += c;      // l'afegim al final de la
sol·licitud
        }
      }
    }

    //Esborrem la variable que permet emmagatzemar la sol·licitud HTTP

    header = "";

    // Tanquem la connexió
```

```
    client.stop();
  }
}
```

B4. Ampliació: zeRGBa (Blynk)

A continuació es mostra el codi amb què s'ha realitzat la primera ampliació de la segona aplicació en mode SoftAP.

zeRGBa.ino

```
// Definim les llibreries necessàries

#define BLYNK_PRINT Serial
#include <ESP8266WiFi.h>
#include <BlynkSimpleEsp8266.h>

// Codi d'autentificació enviat per Blynk a l'e-mail personal

char auth[] = "KIkFIT0sboUF6cTtSheAkv-foLtj1Hdi";

void setup()
{
  Serial.begin(9600);

  // Sincronització de l'app Blynk amb el nostre programa d'Arduino,
gràcies a l'especificació del codi d'autentificació i l'SSID i
contrasenya de la xarxa local

  Blynk.begin(auth, "MiFibra-ACDB", "mYnsjgb5");
}

void loop()
{

  // Execució del programa

  Blynk.run();
}
```

B5. Ampliació: rgbActuator (AllThingsTalk Maker)

RGB - PLACA.ino

A continuació es mostra el codi amb què s'ha realitzat la segona ampliació de la segona aplicació en mode SoftAP.

```
// Definim les llibreries necessàries

#include <AllThingsTalk_WiFi.h> // Llibreria gràcies a la qual podrem
controlar el LED RGB integrat
#include <WiFiNINA.h>           // Necessitem aquesta llibreria per
carregar la biblioteca següent
```

```
#include <utility/wifi_drv.h>   // Exposem les funcions subjacents de
l'MKR1010 per controlar el LED RGB

auto wifiCreds   = WifiCredentials("MiFibra-ACDB", "mYnsjgb5");   //
Definim el nom i la contrasenya de la nostra xarxa local
auto deviceCreds = DeviceConfig("jSPzK8Pyhy0wTlSngfdSpTVi",
"maker:4LQeICzRADFSm0lqFwkEudAztrClyVWDZMHzboE0");       // Introduïm
l'identifcador i el "Token" que AllThingsTalk Maker ens proporciona
auto device      = Device(wifiCreds, deviceCreds);                //
Creem el nostre dispositiu
char* actuator   = "rgbActuator";                                 //
Nom del nostre programa al web AllThingsTalk
int r,g,b;                                                        //
En aquestes variables es mantindran els valors RGB

void setup() {
  Serial.begin(115200);                 // Inicialitzem el port sèrie
a una velocitat de transmissió de 115200 bauds
  WiFiDrv::pinMode(25, OUTPUT);         // Inicialitzem el pin verd
del RGB de l'MKR1010
  WiFiDrv::pinMode(26, OUTPUT);         // Inicialitzem el pin vermell
del RGB de l'MKR1010
  WiFiDrv::pinMode(27, OUTPUT);         // Inicialitzem el pin blau
del RGB de l'MKR1010
  device.debugPort(Serial);             // Habilitem la biblioteca
AllThingsTalk perquè emeti informació pel port sèrie
  device.wifiSignalReporting(true);     // Habilitem la biblioteca
AllThingsTalk perquè des de la web AllThingsTalk Maker es pugui
consultar la força del senyal WiFi rebut
  device.setActuationCallback(actuator, rgb); // Configurem el nostre
programa perquè cada vegada que triem un color des de la web,
s'executi la funció "rgb" que s'encarregarà de donar resposta a la
nostra sol·licitud.
}

void rgb(String value) {                // Funció que es cridarà quan
escollim un color a AllThingsTalk
  Serial.println("Color del RGB canviat!"); // Missatge que veurem pel
port sèrie quan es canviï de color

  // Analitzem la informació RGB rebuda en tres variables diferents
(R, G, B)

  long hexColor = (long) strtol(&value[1], NULL, 16);
  r = hexColor >> 16;
  g = hexColor >> 8 & 0xFF;
  b = hexColor & 0xFF;
  WiFiDrv::analogWrite(25, g);          // Establim el pin verd de
l'RGB segons el valor rebut
  WiFiDrv::analogWrite(26, r);          // Establim el pin vermell de
l'RGB segons el valor rebut
  WiFiDrv::analogWrite(27, b);          // Establim el pin blau de
l'RGB segons el valor rebut
}

void loop() {                           // Executem el programa sempre
que el dispositiu estigui connectat
  device.loop();                        // Mantenim la connexió del
WiFi i AllThingsTalk
}
```

Annex C

El present Annex conté tots els codis finals relatius a la part final pràctica del projecte: *Estació Meteorològica*.

C1. MAC Address ESP32 i ESP8266

A continuació es mostren els codis que permeten obtenir l'adreça MAC de l'ESP32 i l'ESP8266.

MAC_ESP32.ino

```
// Definim les llibreries necessàries

#include <SPI.h>
#include <WiFiNINA.h>

char ssid[] = "MAC";      // Inventem un SSID per a la nostra xarxa

// Estat temporal de la xarxa quan es crida WiFi.beginAP() i es manté
actiu fins que caduca el nombre d'intents (resultant en
WL_CONNECT_FAILED) o fins que s'estableixi una connexió (resultant
WL_CONNECTED)

int status = WL_IDLE_STATUS;

// Variable que emmagatzemarà l'adreça MAC. El 6 indica que contindrà
6 números hexadecimals

byte mac[6];

void setup()
{
 Serial.begin(115200);

 status = WiFi.beginAP(ssid);

 if ( status != WL_AP_LISTENING) {
    Serial.println("No s'ha pogut obtenir una connexió WiFi");
    while(true);
  }

  // Si ens connectem, imprimim la nostre adreça MAC

  else {
  WiFi.macAddress(mac);
  Serial.print("MAC: ");
  Serial.print(mac[0],HEX);
  Serial.print(":");
  Serial.print(mac[1],HEX);
  Serial.print(":");
  Serial.print(mac[2],HEX);
  Serial.print(":");
```

```
  Serial.print(mac[3],HEX);
  Serial.print(":");
  Serial.print(mac[4],HEX);
  Serial.print(":");
  Serial.println(mac[5],HEX);
  }
}

void loop () {}
```

MAC_ESP8266.ino

```
// Definim la llibreria necessària

#include <ESP8266WiFi.h>

void setup() {

  // Inicialitzem el port sèrie

  Serial.begin(115200);
  delay(500);

  Serial.println();
  Serial.print("MAC: ");

  // Declarem la funció que ens permet obtenir la MAC de la nostra
xarxa i l'imprimim pel port sèrie

  Serial.println(WiFi.macAddress());

}

void loop() {}
```

C2. Aplicació final conjunta en mode AP + STA: Estació Meteorològica

A continuació es mostra el codi que ha permès dur a terme l'última aplicació d'aquest projecte.

SERVIDOR - MKR1010.ino

```
// Definim les llibreries necessàries

#include <Scheduler.h> // Aquesta serà l'encarregada d'habilitar les
funcions que permeten la multitasca
#include <WiFiNINA.h>
#include "arduino_secrets.h" // Llibreria definida per nosaltres i que
permet emmagatzemar la contrasenya de la nostra xarxa, per garantir
una mica més de seguretat
#include <Wire.h>
#include <Adafruit_GFX.h> // Aquesta ens permetrà controlar les
funcions de la pantalla OLED
#include <Adafruit_SSD1306.h> // Aquesta ens permetrà controlar les
funcions de la pantalla OLED
```

```
#include "RTClib.h" // Aquesta llibreria permetrà fer servir les
funcions amb què controlarem el mòdul RTC

// Definim l'amplada i alçada de la pantalla, en píxels

#define SCREEN_WIDTH 128 // Amplada pantalla OLED en píxels
#define SCREEN_HEIGHT 64 // Alçada pantalla OLED en píxels

// Inicialitzem la pantalla amb tots els seus paràmetres

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);

// Definim la variable amb què controlarem les operacions del mòdul
RTC

RTC_DS1307 rtc;

char pass[] = SECRET_PASS;

// Definim els dies de la setmana en català

char daysOfTheWeek[7][12] = {"Diumenge", "Dilluns", "Dimarts",
"Dimecres", "Dijous", "Divendres", "Dissabte"};

// Estat temporal de la xarxa quan es crida WiFi.beginAP() i es manté
actiu fins que caduca el nombre d'intents (resultant en
WL_CONNECT_FAILED) o fins que s'estableixi una connexió (resultant
WL_CONNECTED)

int status = WL_IDLE_STATUS;

// Declarem els pins del LED RGB

const int R = 6;      // El pin vermell el connectarem al pin digital 6
de l'Arduino MKR1010
const int G = 7;    // El pin verd el connectarem al pin digital 7 de
l'Arduino MKR1010
const int B = 8;     // El pin blau el connectarem al pin digital 8 de
l'Arduino MKR1010

// Declaració de les variables que emmagatzemen la temperatura i la
humitat dels dos dispositius clients

int tmp_dev1, tmp_dev2;
int hmd_dev1, hmd_dev2;

// Variable que farem servir per fer una interacció de colors quan ens
creem la xarxa

int _step    =  1;
int menuitem = 1;
int page = 1;

// Declaració de variables que farem servir per programar el
funcionament dels 3 polsadors

volatile boolean up = false;
volatile boolean down = false;
volatile boolean middle = false;
int downButtonState = 0;
int upButtonState = 0;
```

```
int selectButtonState = 0;
int lastDownButtonState = 0;
int lastSelectButtonState = 0;
int lastUpButtonState = 0;

// Variables que contindran el valor de la temperatura i la humitat de
l'URL entrant dels clients

String value1;
String value2;

// Obrim el port 80 (port per a transmissions HTTP) del nostre
encaminador, perquè el servidor es pugui comunicar amb els clients

WiFiServer server(80);

// Creem un parell de funcions que executaran una transició de colors
quan es creï la xarxa

int FadeOn(int pin)
{
  int fadeValue;
  // Ramp up pin
  for (fadeValue = 0 ; fadeValue <= 255; fadeValue += _step) {
    // sets the value (range from 0 to 255):
    analogWrite(pin, fadeValue);
    // wait to see the dimming effect
    delay(6);
  }
  analogWrite(pin, 255);
  return (fadeValue);
}
int FadeOff(int pin)
{
  int fadeValue;
  // Ramp down pin
  for (fadeValue = 255 ; fadeValue >= 0; fadeValue -= _step) {
    // sets the value (range from 0 to 255):
    analogWrite(pin, fadeValue);
    // wait to see the dimming effect
    delay(6);
  }
  analogWrite(pin, 0);
  return (fadeValue);
}

void setup() {

  // Definim els pins del LED RGB com a sortides

  pinMode(R, OUTPUT);
  pinMode(G, OUTPUT);
  pinMode(B, OUTPUT);

  // Cridem la funció que ens permetrà configurar i inicialitzar el
mòdul WiFi

  setupWiFi();

  // Inicialitzem el servidor
```

```
  server.begin();

  // Iniciem la pantalla OLED amb l'adreça 0x78 corresponent a la seva
direcció I2C

  display.begin(SSD1306_SWITCHCAPVCC, 0x78 >> 1);

  // Configurem les pantalles de visualització de l'SSD1306

  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  //display.setTextColor(BLACK, WHITE);
  display.setCursor(0, 0);
  display.println("   ESP32");
  display.setTextSize(3);
  //display.setTextColor(BLACK, WHITE);
  display.setTextColor(WHITE);
  display.setCursor(10, 20);
  display.println("AP+STA");
  display.setTextSize(1);
  display.setTextColor(WHITE);
  //display.setTextColor(BLACK, WHITE);
  display.setCursor(0, 52);
  display.println("Treball de Fi de Grau");

  display.display();
  delay(5000);

  display.clearDisplay();
  display.setTextSize(2);
  display.setTextColor(WHITE);
  //display.setTextColor(BLACK, WHITE);
  display.setCursor(0, 0);
  display.println("Connectant");
  display.clearDisplay();
  display.setTextSize(1);       display.setTextColor(WHITE);

  // Obtenim l'adreça MAC del nostre dispositiu, gràcies a la qual
donarem el nom a la nostra xarxa quan definim l'SSID

  uint8_t mac[WL_MAC_ADDR_LENGTH];
  WiFi.macAddress(mac);
  String macID = String(mac[WL_MAC_ADDR_LENGTH - 2], HEX) +
String(mac[WL_MAC_ADDR_LENGTH - 1], HEX);
  macID.toUpperCase();
  display.setCursor(0, 0);       display.print("SSID:");
display.println("TFG - ESP32" + macID);
  display.setTextSize(2);       display.setTextColor(WHITE);
  display.setCursor(0, 18);      display.println(WiFi.localIP());
  display.display();
  analogWrite(R, 0);
  analogWrite(G, 0);

  // Un cop finalitzada la creació de la xarxa, es procedeix a la
visualització de transicions de colors de l'RGB

  FadeOn(B);
  FadeOn(R);
  FadeOff(B);
  FadeOn(G);
```

```
  FadeOff(R);
  FadeOn(B);
  FadeOff(G);
  analogWrite(R, 0);
  analogWrite(G, 0);
  analogWrite(B, 0);

  // Iniciem la segona tasca que farà el programa. Aquesta és la de
fer funcionar el mòdul RTC

  Scheduler.startLoop(loop2);

  if (! rtc.begin()) {
    Serial.println("No hi ha mòdul RTC");
    while (1);
  }
  // La instrucció que veurem comentada a continuació serveix per
posar en hora el mòdul RTC la primera vegada que carreguem el
programa. Després aquesta funció ja no cal i es deixa en forma de
comentari
  // rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));
}


void loop() {

  // Funció que ens permet donar el disseny al nostre menú

  drawMenu();

  // Indiquem on estan connectats físicament els pins dels polsadors

  downButtonState = digitalRead(0);
  selectButtonState = digitalRead(1);
  upButtonState =   digitalRead(4);

  // Un cop indicat, es procedeix  a l'execució de la funció que
detecta si es prem cap dels tres polsadors

  checkIfDownButtonIsPressed();
  checkIfUpButtonIsPressed();
  checkIfSelectButtonIsPressed();

  // Declarem el comportament que ha de tenir el menú segons el
polsador que es premi

  if (up && page == 1 ) {
    up = false;
    menuitem--;
    if (menuitem == 0)
    {
      menuitem = 3;
    }
  }
  if (down && page == 1) {
    down = false;
    menuitem++;
    if (menuitem == 4)
    {
      menuitem = 1;
    }
```

```
  }
  if (middle) {
    middle = false;
    if (page == 1 && menuitem == 1) {
      page = 2;
    }
    else if (page == 2) {
      page = 1;
    }
    if (page == 1 && menuitem == 2) {
      page = 3;
    }
    else if (page == 3) {
      page = 1;
    }
    if (page == 1 && menuitem == 3) {
      page = 4;
    }
    else if (page == 4) {
      page = 1;
    }
  }

  // Definim el comportament del LED RGB en base a la temperatura que
es llegeix

  if (page == 2) {
    if (tmp_dev1 >= 28) {
      analogWrite(R, 255);
      analogWrite(G, 0);
      analogWrite(B, 0);
    }
    else if (tmp_dev1 >= 24 && tmp_dev1 <= 27) {
      analogWrite(R, 255);
      analogWrite(G, 255);
      analogWrite(B, 0);
    }
    else if (tmp_dev1 >= 18 && tmp_dev1 <= 23) {
      analogWrite(G, 255);
      analogWrite(R, 0);
      analogWrite(B, 0);
    }
    else  if (tmp_dev1 >= 14 && tmp_dev1 <= 17) {
      analogWrite(G, 255);
      analogWrite(B, 255);
      analogWrite(R, 0);
    }
    else if (tmp_dev1 <= 13) {
      analogWrite(B, 255);
      analogWrite(R, 0);
      analogWrite(G, 0);
    }
  }
  else if (page == 3) {
    if (tmp_dev2 >= 28) {
      analogWrite(R, 255);
      analogWrite(G, 0);
      analogWrite(B, 0);
    }
    else if (tmp_dev2 >= 24 && tmp_dev2 <= 27) {
      analogWrite(R, 255);
```

```
      analogWrite(G, 255);
      analogWrite(B, 0);
    }
    else if (tmp_dev2 >= 18 && tmp_dev2 <= 23) {
      analogWrite(G, 255);
      analogWrite(B, 0);
      analogWrite(R, 0);
    }
    else if (tmp_dev2 >= 14 && tmp_dev2 <= 17) {
      analogWrite(G, 255);
      analogWrite(B, 255);
      analogWrite(R, 0);
    }
    else if (tmp_dev2 <= 13) {
      analogWrite(B, 255);
      analogWrite(R, 0);
      analogWrite(G, 0);
    }
  }
  else {
    analogWrite(R, 0);
    analogWrite(G, 0);
    analogWrite(B, 0);
  }
}


void decoder_values(String _req) {

  // Es descodifiquen les dues variables "value" que ens envien els
dispositius ESP8266

  int Start1 = _req.indexOf("=");
  int Finish1 = _req.indexOf('&', Start1 + 1);
  int Start2 = _req.indexOf("=", Finish1 + 1);
  int Finish2 = _req.indexOf("/", Start2 + 1);

  value1 = "";
  value2 = "";

  for (int i = Start1 + 1; i < Finish1; i++)
  {
    value1 = value1 + _req.charAt(i);
  }
  for (int i = Start2 + 1; i < Finish2; i++)
  {
    value2 = value2 + _req.charAt(i);
  }
}

// Funció que havíem declarat abans i que s'encarrega de crear la
xarxa i donar el nom SSID d'aquesta

void setupWiFi() {
  uint8_t mac[WL_MAC_ADDR_LENGTH];
  WiFi.macAddress(mac);
  String macID = String(mac[WL_MAC_ADDR_LENGTH - 2], HEX) +
String(mac[WL_MAC_ADDR_LENGTH - 1], HEX);
  macID.toUpperCase();
  String AP_NameString = ("TFG - ESP32" + macID);
```

```
  char AP_NameChar[AP_NameString.length() + 1];
  memset(AP_NameChar, AP_NameString.length() + 1, 0);

  for (int i = 0; i < AP_NameString.length(); i++)
    AP_NameChar[i] = AP_NameString.charAt(i);

  // Inicialitzem el port sèrie

  Serial.begin(115200);

  Serial.println("Mode softAP actiu");

  // Comprovem l'estat del mòdul WiFi

  if (WiFi.status() == WL_NO_MODULE) {
    Serial.println("El mode softAP no s'ha pogut activar!");

    // Si no s'ha pogut trobar cap client, el programa no continua

    while (true);
  }

  // Creació de la nostra xarxa

  status = WiFi.beginAP(AP_NameChar, pass);
  if (status != WL_AP_LISTENING) {
    Serial.println("No s'ha pogut crear una xarxa en mode AP");

    // Si no s'ha pogut crear la xarxa, el programa no continua

    while (true);
  }

  // Esperem 10 segons perquè el client es connecti

  delay(10000);
}

// A continuació es declaren un conjunt de funcions que havíem cridat
anteriorment i que permeten comprovar si s'ha premut cap polsador

void checkIfDownButtonIsPressed() {
  if (downButtonState != lastDownButtonState) {
    if (downButtonState == 0) {
      down = true;
    }
    delay(50);
  }
  lastDownButtonState = downButtonState;
}

void checkIfSelectButtonIsPressed() {
  if (upButtonState != lastUpButtonState) {
    if (upButtonState == 0) {
      up = true;
    }
    delay(50);
  }
  lastUpButtonState = upButtonState;
}
```

```
void checkIfUpButtonIsPressed() {
  if (selectButtonState != lastSelectButtonState)
  {
    if (selectButtonState == 0) {
      middle = true;
    }
    delay(50);
  }
  lastSelectButtonState = selectButtonState;
}

// Dissenyem el nostre menú

void drawMenu() {
  String s;
  WiFiClient client = server.available();
  if (!client) {
    return;
  }
  String req = client.readStringUntil('\r');
  Serial.print(req);

  client.flush();

  int val = -1;

  if (page == 1) {
    display.setTextSize(2);
    display.clearDisplay();
    display.setTextColor(WHITE, BLACK);
    display.setCursor(40, 0);
    display.print("MENU");
    display.drawFastHLine(14, 15, 100, WHITE);
    display.setCursor(14, 20);

    if (menuitem == 1) {
      display.setTextColor(BLACK, WHITE);
    }
    else {
      display.setTextColor(WHITE, BLACK);
    }
    display.setTextSize(1);
    display.print(">LOLIN BLAU");
    display.setCursor(14, 35);
    if (menuitem == 2) {
      display.setTextColor(BLACK, WHITE);
    }
    else {
      display.setTextColor(WHITE, BLACK);
    }
    display.setTextSize(1);
    display.print(">LOLIN GROC");
    if (menuitem == 3) {
      display.setTextColor(BLACK, WHITE);
    }
    else {
      display.setTextColor(WHITE, BLACK);
    }
    display.setTextSize(1);
    display.setCursor(14, 50);
    display.print(">DATA I HORA");
```

```
    display.display();
  }

  else if (page == 2) {
    if (req.indexOf("/158e") != -1) {
      val = 0;
      decoder_values(req);
      tmp_dev1 = value1.toInt();
      hmd_dev1 = value2.toInt();
      Serial.println("");
      Serial.print("Temperatura_dev1: ");
      Serial.println(tmp_dev1);
      Serial.print("Humedad_dev1: ");
      Serial.println(hmd_dev1);

      display.clearDisplay();
      display.setTextSize(1);
      display.setTextColor(WHITE, BLACK);
      display.setCursor(10, 3);
      display.print("EST. METEOROLOGICA");
      display.drawRect(0, 0, 128, 64, WHITE);
      display.drawLine(0, 13, 128, 13, WHITE);
      display.display();
      display.setCursor(5, 25);
      display.print("TEMP: "); display.print(tmp_dev1);
display.drawRect(55, 22, 3, 3, WHITE); display.print(" C");
      display.print("  HUM: "); display.print(hmd_dev1);
display.print("%");
      display.setCursor(24, 40);
      display.print("> LOLIN BLAU <");

      display.display();
    }
  }
  else if (page == 3) {
    if (req.indexOf("/54fe") != -1) {
      decoder_values(req);
      tmp_dev2 = value1.toInt();
      hmd_dev2 = value2.toInt();
      Serial.println("");
      Serial.print("Temperatura_dev2: ");
      Serial.println(tmp_dev2);
      Serial.print("Humedad_dev2: ");
      Serial.println(hmd_dev2);

      display.clearDisplay();
      display.setTextSize(1);
      display.setTextColor(WHITE, BLACK);
      display.setCursor(10, 3);
      display.print("EST. METEOROLOGICA");
      display.drawRect(0, 0, 128, 64, WHITE);
      display.drawLine(0, 13, 128, 13, WHITE);
      display.display();
      display.setCursor(5, 25);
      display.print("TEMP: "); display.print(tmp_dev2);
display.drawRect(55, 22, 3, 3, WHITE); display.print(" C");
      display.print("  HUM: "); display.print(hmd_dev2);
display.print("%");
      display.setCursor(24, 40);
      display.print("> LOLIN GROC <");
```

```
      display.display();
    }
  }
}

// Acció paral·lela que executarà el nostre programa, encarregada de
donar la data i l'hora

void loop2() {
  if (page == 4) {
    DateTime now = rtc.now();

    int hora = now.hour();
    int minuts = now.minute();
    int load = 0;
    uint8_t parell;

    display.clearDisplay();
    display.fillRoundRect(0, 0, 128, 12, 3, WHITE);
    display.setCursor(8, 2);
    // Tamaño del texto
    display.setTextSize(1);
    display.setTextColor(SSD1306_BLACK);
    display.print(daysOfTheWeek[now.dayOfTheWeek()]);
    display.print("  ");
    display.print(now.day());
    display.print("/");
    display.print(now.month());
    display.print("/");
    display.println(now.year());
    display.setTextColor(WHITE);
    // Tamaño del texto
    display.setTextSize(3);
    display.setCursor(15, 20);

    display.print(hora);
    parell = now.second() & 0b1;
    if (parell)
    {
      //Número imparell

      display.print(" ");
    }
    else
    {
      //Número parell

      display.print(":");
    }
    if (minuts < 10)
    {
      display.print("0");
    }
    else
    {
      // Posició del text

      display.print("");
    }
    display.print(minuts);
```

```
    display.drawCircle(117, 48, 10, WHITE);
    display.setCursor(112, 45);
    display.setTextSize(1);
    display.setTextColor(WHITE);
    if (now.second() < 10)
    {
      display.print("0");
      display.print(now.second());
    }
    else
    {
      display.println(now.second());
    }
    load = map(now.second(), 0, 59, 15, 106);
    display.drawLine(15, 50, load, 50, WHITE);

    // Mostrem per pantalla

    display.display();

    // Netegem la pantalla

    display.clearDisplay();
  }

  // Comanda que s'encarrega de fer funcionar correctament la
multitasca

  yield();
}
```

CLIENT – LOLIN_D1_mini.ino

```
// Definim les llibreries necessàries

#include <ESP8266WiFi.h>
#include "DHT.h" // Llibreria que gestionarà les dades del sensor
mitjançant una sincronització precisa
#define DHTTYPE DHT11   // Existeixen diversos models de sensor DHT.
Per tant, especifiquem el que farem servir

// Definim l'SSID i la contrasenya del servidor al qual ens volem
connectar (MKR1010)

const char WiFiSSID[] = "TFG - ESP32CFA4";
const char WiFiPSK[] = "87654321";

// Declarem l'adreça IP i el port d'enllaç del nostre servidor

const char host[] = "192.168.4.1";
const int httpPort = 80;

String macID; //String on s'emmagatzemarà la nostra macID

//Variables on s'emmagatzemaran les mesures de la temperatura i
humitat

int value1;
```

```
int value2;

// Inicialització dels pins

const int LED_PIN = D5;
uint8_t DHTPin = D4;

// Instal·lem la llibreria per controlar el sensor DHT11

DHT dht(DHTPin, DHTTYPE);

void setup() {
  initHardware(); // Funció en la qual inicialitzarem i configurarem
el port sèrie i configurarem el nostre LED
}

void loop() {
  connectWiFi();  // Funció que s'encarregarà de la connexió WiFi
  delay(6000);  //Esperem sis segons a relitzar una nova lectura del
sensor DHT11
  float temperature = dht.readTemperature();
  float humidity = dht.readHumidity();
  float bTemperature;
  float bHumidity;

  // Si la lectura és errònia, no l'enviem al servidor

  if (!isnan(temperature || humidity)) {
    bTemperature = temperature;
    bHumidity = humidity;
    Serial.print(bTemperature); Serial.println("°C");
    Serial.print(bHumidity); Serial.println("%");
  }
  else if (isnan(temperature || humidity)) {
    temperature = bTemperature;
    humidity = bHumidity;
    Serial.print(temperature); Serial.println("°C");
    Serial.print(humidity); Serial.println("%");
  }

  // Emmagatzemem en les variables de sortida el valor llegit de la
temperatura i humitat

  value1 = (int)temperature;
  value2 = (int)humidity;

  // Iniciem la connexió amb el servidor

  Serial.print("Connectant a ");
  Serial.println(host);

  // S'intenta la connexió amb el servidor

  WiFiClient client;
  if (!client.connect(host, httpPort)) {
    Serial.println("Connexió fallida");
    return;
  }
```

```
  // Si ha existit connexió, es continua i es concatena el macID dels
nostres ESP8266 amb els valors de temperatura i humitat llegits, per
convertir-se en un URL

  String url = macID;
  url += "&value1=";
  url += value1;
  url += "&value2=";
  url += value2;

  Serial.print("URL enviada: ");
  Serial.println(url);

  // S'envia la sol·licitud al servidor

  client.print(String("GET /") + url + "/ HTTP/1.1\r\n");

  unsigned long timeout = millis();
  while (client.available() == 0) {
    if (millis() - timeout > 5000) {
      Serial.println(">>> Nou enviament de dades");
      client.stop();
      return;
    }
  }

  // Es llegeix tot el que es rep des del servidor i s'imprimeix al
port sèrie

  while (client.available()) {
    String line = client.readStringUntil('\r');
    Serial.print(line);
  }

  Serial.println();
  Serial.println("Tancant connexió");
}

void connectWiFi()
{

  // Obtenim la macID del nostre ESP8266

  uint8_t mac[WL_MAC_ADDR_LENGTH];
  WiFi.macAddress(mac);
  macID = String(mac[WL_MAC_ADDR_LENGTH - 2], HEX) +
String(mac[WL_MAC_ADDR_LENGTH - 1], HEX);
  Serial.print(macID);

  // A continuació ens connectem a la xarxa WiFi del servidor

  Serial.println();
  Serial.println();
  Serial.print("Connectant a ");
  Serial.println(WiFiSSID);

  // Configurem el nostre ESP8266 com a STA

  WiFi.mode(WIFI_STA);

  // S'inicia la connexió al nostre servidor
```

```
  WiFi.begin(WiFiSSID, WiFiPSK);
  byte ledStatus = LOW;

  // Esperem a estar connectats

  while (WiFi.status() != WL_CONNECTED)
  {
    // Mentre no ho estem, el LED fa pampallugues

    Serial.print(".");
    digitalWrite(LED_PIN, ledStatus);

    // Quan ens connectem, el LED queda encès

    ledStatus = (ledStatus == HIGH) ? LOW : HIGH;
    delay(100);
  }
  Serial.println("WiFi connectat");
  Serial.println("Adreça IP del client: ");
  Serial.println(WiFi.localIP());
}

void initHardware()
{
  Serial.begin(115200); //Iniciem el port sèrie
  pinMode(LED_PIN, OUTPUT); // Configurem el nostre LED
  digitalWrite(LED_PIN, HIGH);
}
```