Proyecto Final:

Código:

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <WiFi.h>
#include <time.h>
#include "Adafruit_I2CDevice.h"
#include "ESPAsyncWebServer.h"


//Puerto 80
WiFiServer server(80);

// Wifi
const char* ssid     = "Telefedu";
const char* password = "3e6eb1e28071";

//Variables
int contconexion = 0;

String header; // HTTP Request

//HTML
String paginaInicio = "<!DOCTYPE html>"
"<html>"
"<head>"
"<meta charset='utf-8' />"
"<META HTTP-EQUIV='Refresh' CONTENT='1'>"
"<title> Informacion Actual </title>"
"</head>"
"<body>"
"<center>"
"<h1> <u> Informacion Actual </u> </h1>"
"<br>"
"</center>";


String Temperatura =
"<h2> <ul> <li> Temperature: </li> </ul> </h2>"
"<h2><h2>";

String Humedad =
"<h2> <ul> <li> Humidity: </li> </ul> </h2>"
"<h2><h2><br>";

String Presion =
"<h2> <ul> <li> Pressure: </li> </ul> </h2>"
"<h2><h2><br>";

String paginaFin =
"</body>"
"</html>";

// NTP Server
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 0;
const int   daylightOffset_sec = 7200;
```

```
const int   daylightOffset_sec = 7200;

//Display
#define SCREEN_WIDTH 128  // OLED display width, in pixels
#define SCREEN_HEIGHT 64  // OLED display height, in pixels

#define I2Cdisplay_SDA 21
#define I2Cdisplay_SCL 22
TwoWire I2Cdisplay = TwoWire(1);

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &I2Cdisplay, -1);

// BME280
#define I2C_SDA 21
#define I2C_SCL 22
TwoWire I2CBME = TwoWire(0);
Adafruit_BME280 bme;

//Pantallas
int displayScreenNum = 0;
int displayScreenNumMax = 3;

unsigned long lastTimer = 0;
unsigned long timerDelay = 7500;


//Circulitos
void displayIndicator(int displayNumber) {
  int xCoordinates[5] = {49, 59, 69, 79};
  for (int i =0; i<4; i++) {
    if (i == displayNumber) {
      display.fillCircle(xCoordinates[i], 60, 2, WHITE);
    }
    else {
      display.drawCircle(xCoordinates[i], 60, 2, WHITE);
    }
  }
}

//Fecha y Hora
void displayLocalTime(){
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");

  //GET DATE
  //Get full weekday name
  char weekDay[10];
  strftime(weekDay, sizeof(weekDay), "%a", &timeinfo);
  //Get day of month
  char dayMonth[4];
  strftime(dayMonth, sizeof(dayMonth), "%d", &timeinfo);
  //Get abbreviated month name
  char monthName[5];
  strftime(monthName, sizeof(monthName), "%b", &timeinfo);
  //Get year
  char year[6];
  strftime(year, sizeof(year), "%Y", &timeinfo);

  //GET TIME
  //Get hour (12 hour format)
  /*char hour[4];
  strftime(hour, sizeof(hour), "%I", &timeinfo);*/
```

```
  //Get hour (24 hour format)
  char hour[4];
  strftime(hour, sizeof(hour), "%H", &timeinfo);
  //Get minute
  char minute[4];
  strftime(minute, sizeof(minute), "%M", &timeinfo);

  //Display Date and Time on OLED display
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(3);
  display.setCursor(19,5);
  display.print(hour);
  display.print(":");
  display.print(minute);
  display.setTextSize(1);
  display.setCursor(16,40);
  display.print(weekDay);
  display.print(", ");
  display.print(dayMonth);
  display.print(" ");
  display.print(monthName);
  display.print(" ");
  display.print(year);
  displayIndicator(displayScreenNum);
  display.display();
}

//Temperatura
void displayTemperature(){
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 5);
  display.print("Temperature ");
  display.setTextSize(3);
  display.setCursor(15, 25);
  delay(200);
  float temperature = bme.readTemperature();
  temperature = bme.readTemperature();
  display.print(temperature);
  display.cp437(true);
  display.setTextSize(1);
  display.print(" ");
  display.write(167);
  display.print("C");
  display.setCursor(0, 34);
  display.setTextSize(1);
  displayIndicator(displayScreenNum);
  display.display();
}

//Humedad
void displayHumidity(){
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 5);
  display.print("Humidity ");
  display.setTextColor(WHITE);
  display.setTextSize(3);
  display.setCursor(15, 25);
  float humidity = bme.readHumidity();
  display.print(humidity);
  display.setTextSize(1);
  display.print(" %");
  display.setCursor(0, 34);
```

```
  display.setTextSize(1);
  displayIndicator(displayScreenNum);
  display.display();
}

//Presion
void displayPressure(){
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 5);
  display.print("Pressure ");
  display.setTextSize(2);
  display.setCursor(15, 25);
  display.print(bme.readPressure()/100.0F);
  display.setTextSize(1);
  display.print(" hpa");
  display.setCursor(0, 34);
  display.setTextSize(1);
  displayIndicator(displayScreenNum);
  display.display();
}

//Subprograma encargado de mostar la pantalla correcta
void updateScreen() {
  if (displayScreenNum == 0){
    displayLocalTime();
  }
  else if (displayScreenNum == 1) {
    displayTemperature();
  }
  else if (displayScreenNum ==2){
    displayHumidity();
  }
  else {
    displayPressure();
  }
}

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

  I2CBME.begin(I2C_SDA, I2C_SCL, 100000);
  I2Cdisplay.begin(I2Cdisplay_SDA, I2Cdisplay_SCL, 100000);

  //Inicializar Display
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  display.clearDisplay();
  display.setTextColor(WHITE);

  //Inicializar BME280
  bool status = bme.begin(0x76, &I2CBME);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  //Wi-Fi
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  //Cuenta hasta 50 si no se puede conectar lo cancela
  while (WiFi.status() != WL_CONNECTED and contconexion <50) {
```

```
  while (WiFi.status() != WL_CONNECTED and contconexion <50) {
    ++contconexion;
    delay(500);
    Serial.print(".");
  }

  if (contconexion <50) {
      Serial.println("");
      Serial.println("WiFi conectado");
      Serial.println(WiFi.localIP());
      server.begin(); // iniciamos el servidor
  }
  else {
      Serial.println("");
      Serial.println("Error de conexion");
  }

  // Init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}

void loop() {

  // Change screen every 7.5 seconds (timerDelay variable)
  if ((millis() - lastTimer) > timerDelay) {
    updateScreen();
    Serial.println(displayScreenNum);
    if(displayScreenNum < displayScreenNumMax) {
      displayScreenNum++;
    }
    else {
      displayScreenNum = 0;
    }
    lastTimer = millis();
  }

  //Pagina Web
  WiFiClient client = server.available();   // clientes entrantes

  if (client) {                             // Si se conecta
    Serial.println("New Client.");          //
    String currentLine = "";                //
    while (client.connected()) {            // loop mientras el cliente est谩 conectado
      if (client.available()) {             // si hay bytes para leer desde el cliente
        char c = client.read();             // lee un byte
        Serial.write(c);                    //
        header += c;
        if (c == '\n') {                    // si el byte es un caracter de salto de linea
          // HTTP request del cliente, entonces respondemos:
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Mostrar pagina web
            float temperature = bme.readTemperature();
            float humidity = bme.readHumidity();
            float pressure = bme.readPressure()/100.0F;
            client.println(paginaInicio + Temperatura + String(temperature) + Humedad + String(humidity) + Presion + String(pressure

            // la respuesta HTTP temina
            client.println();
            break;
          } else { // si tenemos una nueva linea limpiamos
            currentLine = "";
```

```
          }
        } else if (c != '\r') {  // si C es distinto al caracter de retorno
          currentLine += c;      // lo agrega al final de currentLine
        }
      }
    }
    // Limpiamos
    header = "";
    // Cerrar Conexion
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

Explicacion del codigo:

Como con todo programa, antes de nada tenemos que declarar todas las librerias que sean necessarias, en nuestro caso algunas hay que instalarlas y otras no hace falta, aun asi todas son importantes. Las librerias son:

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <Adafruit_Sensor.h>
#include <Adafruit_BME280.h>
#include <WiFi.h>
#include <time.h>
#include "Adafruit_I2CDevice.h"
#include "ESPAsyncWebServer.h"
```

Seguidamente, para poder connectar la placa ESP32 a una red Wi-Fi tenemos que declarar dos variables, el nombre de la red WiFi y la contraseña de esta:

```
const char* ssid     = "Telefedu";
const char* password = "3e6eb1e28071";
```

Ahora iniciamos una variable que sera un contador, tambies hacemos una variable para guradar el HTTP request. Seguidamente ponemos el código HTML, el cual mostrara todo lo que se vera en la pagina web, incluido los datos que captara nuestro sensor:

```
String paginaInicio = "<!DOCTYPE html>"
"<html>"
"<head>"
"<meta charset='utf-8' />"
"<META HTTP-EQUIV='Refresh' CONTENT='1'>"
"<title> Informacion Actual </title>"
"</head>"
"<body>"
"<center>"
"<h1> <u> Informacion Actual </u> </h1>"
"<br>"
"</center>";


String Temperatura =
"<h2> <ul> <li> Temperature: </li> </ul> </h2>"
"<h2><h2>";

String Humedad =
"<h2> <ul> <li> Humidity: </li> </ul> </h2>"
"<h2><h2><br>";

String Presion =
"<h2> <ul> <li> Pressure: </li> </ul> </h2>"
"<h2><h2><br>";

String paginaFin =
"</body>"
"</html>";
```

Ahora, definiremos varias variables que serviran para tener la hora correctamente ya que solicitamos la hora de un servidor NTP, asignamos la hora a GTM y finalmente vemos a cuanto corresponde en horario de España. Lo cual nos dice que 2h son 7200 segundos.

```
const char* ntpServer = "pool.ntp.org";
const long  gmtOffset_sec = 0;
const int   daylightOffset_sec = 7200;
```

A continuacion definimos las medidas de nuestro display:

```
#define SCREEN_WIDTH 128  // OLED display width, in pixels
#define SCREEN_HEIGHT 64  // OLED display height, in pixels
```

Ahora ssignamos los pines para el dispositivo I2C: - SDA --> GPIO 22 - SCL --> GPIO 21

```
#define I2Cdisplay_SDA 21
#define I2Cdisplay_SCL 22
TwoWire I2Cdisplay = TwoWire(1);
```

Inicializamos un monitor con un protocolo de communicación I2C:

```
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &I2Cdisplay, -1);
```

Asignamos los pines que vamos a usar para conectar el dispositivo I2C: - SDA --> GPIO 22 - SCL --> GPIO 21

También creamos un objeto que llamaremos bme con el Adafruit_BME280:

```
#define I2C_SDA 21
#define I2C_SCL 22
TwoWire I2CBME = TwoWire(0);
Adafruit_BME280 bme;
```

Para poder manejar los contadores definimos dos nuevas variables:

```
unsigned long lastTimer = 0;
unsigned long timerDelay = 7500;
```

Ahora nos encontramos nuestro primer subprograma cuyo trabajo es mostrar unas pequeñas circumeferncias debajo del display para saber en que pantalla nos encontramos en ese preciso momento:

```
void displayIndicator(int displayNumber) {
  int xCoordinates[5] = {49, 59, 69, 79};
  for (int i =0; i<4; i++) {
    if (i == displayNumber) {
      display.fillCircle(xCoordinates[i], 60, 2, WHITE);
    }
    else {
      display.drawCircle(xCoordinates[i], 60, 2, WHITE);
    }
  }
}
```

Ahora veremos este subprograma el cual nos muestra la primera pantalla del display,la cual mostrara hora, dia y fecha.

```
void displayLocalTime(){
  struct tm timeinfo;
  if(!getLocalTime(&timeinfo)){
    Serial.println("Failed to obtain time");
  }
  Serial.println(&timeinfo, "%A, %B %d %Y %H:%M:%S");

  //GET DATE
  //Get full weekday name
  char weekDay[10];
  strftime(weekDay, sizeof(weekDay), "%a", &timeinfo);
  //Get day of month
  char dayMonth[4];
  strftime(dayMonth, sizeof(dayMonth), "%d", &timeinfo);
  //Get abbreviated month name
  char monthName[5];
  strftime(monthName, sizeof(monthName), "%b", &timeinfo);
  //Get year
  char year[6];
  strftime(year, sizeof(year), "%Y", &timeinfo);

  //GET TIME
  //Get hour (12 hour format)
  /*char hour[4];
  strftime(hour, sizeof(hour), "%I", &timeinfo);*/

  //Get hour (24 hour format)
  char hour[4];
  strftime(hour, sizeof(hour), "%H", &timeinfo);
  //Get minute
  char minute[4];
  strftime(minute, sizeof(minute), "%M", &timeinfo);

  //Display Date and Time on OLED display
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(3);
  display.setCursor(19,5);
  display.print(hour);
  display.print(":");
  display.print(minute);
  display.setTextSize(1);
  display.setCursor(16,40);
  display.print(weekDay);
  display.print(", ");
  display.print(dayMonth);
  display.print(" ");
  display.print(monthName);
  display.print(" ");
  display.print(year);
  displayIndicator(displayScreenNum);
  display.display();
}
```

El siguiente subprograma se encarga de la segunda pantalla del display la cual nos mostrara la tempereatura ambiente que capta el sensor:

```
void displayTemperature(){
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 5);
  display.print("Temperature ");
  display.setTextSize(3);
  display.setCursor(15, 25);
  delay(200);
  float temperature = bme.readTemperature();
  temperature = bme.readTemperature();
  display.print(temperature);
  display.cp437(true);
  display.setTextSize(1);
  display.print(" ");
  display.write(167);
  display.print("C");
  display.setCursor(0, 34);
  display.setTextSize(1);
  displayIndicator(displayScreenNum);
  display.display();
}
```

Este subprograma se encarga de la segunda pantalla del display y muestra la humedad que capta el sensor:

```
void displayHumidity(){
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 5);
  display.print("Humidity ");
  display.setTextColor(WHITE);
  display.setTextSize(3);
  display.setCursor(15, 25);
  float humidity = bme.readHumidity();
  display.print(humidity);
  display.setTextSize(1);
  display.print(" %");
  display.setCursor(0, 34);
  display.setTextSize(1);
  displayIndicator(displayScreenNum);
  display.display();
}
```

Finalmente este subprograma se encarga de mostrar por la pantalla del display la presión que capta el sensor:

```
void displayPressure(){
  display.clearDisplay();
  display.setTextSize(1);
  display.setCursor(0, 5);
  display.print("Pressure ");
  display.setTextSize(2);
  display.setCursor(15, 25);
  display.print(bme.readPressure()/100.0F);
  display.setTextSize(1);
  display.print(" hpa");
  display.setCursor(0, 34);
  display.setTextSize(1);
  displayIndicator(displayScreenNum);
  display.display();
}
```

Ahora este subprograma se encarara de ir actualizando las pantallas y comprovar que se muestre la correcta, para ello usaremos unos if ese de manera que siempre hay una pantalla mostrandose:

```
void updateScreen() {
  if (displayScreenNum == 0){
    displayLocalTime();
  }
  else if (displayScreenNum == 1) {
    displayTemperature();
  }
  else if (displayScreenNum ==2){
    displayHumidity();
  }
  else {
    displayPressure();
  }
}
```

Ahora llegamos al setup, el cual se encargara de todas las inicializaciones del programa.
Primeramente nos encontramos con unserial.begin el cual abre el puerto serie y fija la velocidad para la transmisión de datos, esta velocidad es la indicada en el monitor_speed del .ini, seguidamente nos encontramos con la iniciacion de varios componentes de nuestro proyecto, como el bme280 y el display i2c ssd1306, tambien tenemos la comprovacion de que estos estan funcionando, ya que sino fuera asi, el programa nos daria un aviso de ello.
Seguidamente una vez hechas las comprobaciones y verificar que tenemos conexión nos encontramos con el codigo que nos conectara la esp32 a la red wifi, y de la misma manera que antes, en caso de que haya un fallo de conexion el mismo programa nos avisaria con un mensaje.
Por último vemos el codigo configTime el cual usamos para sincronizar la hora.

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

  I2CBME.begin(I2C_SDA, I2C_SCL, 100000);
  I2Cdisplay.begin(I2Cdisplay_SDA, I2Cdisplay_SCL, 100000);

  // Initialize OLED Display
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for(;;);
  }
  display.clearDisplay();
  display.setTextColor(WHITE);

  // Initialize BME280
  bool status = bme.begin(0x76, &I2CBME);
  if (!status) {
    Serial.println("Could not find a valid BME280 sensor, check wiring!");
    while (1);
  }

  // Connect to Wi-Fi
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  //Cuenta hasta 50 si no se puede conectar lo cancela
  while (WiFi.status() != WL_CONNECTED and contconexion <50) {
    ++contconexion;
    delay(500);
    Serial.print(".");
  }

  if (contconexion <50) {
      Serial.println("");
      Serial.println("WiFi conectado");
      Serial.println(WiFi.localIP());
      server.begin(); // iniciamos el servidor
  }
  else {
      Serial.println("");
      Serial.println("Error de conexion");
  }

  // Init and get the time
  configTime(gmtOffset_sec, daylightOffset_sec, ntpServer);
}
```

Finalmente llegamos al loop este se compone por varios bloques de instrucciones, primeramente nos encontramos el "if" encargado de cambiar de pantalla cada 7.5 segundos, esto lo hace sumando todos los milisegundos hasta llegar al numero que hemos establecido anteriormente.
A continuacion nos enconramos con el WifiClient, el cual nos dice si el cliente esta disponible ono, y es el encargado de mostrarnos la pagina creada anteriormente en HTML, de manera su funcion tambien es que se puedan ver los datos que lee nuestro sensor. Ademas, esta pagina ha sido hecha con un tiempo de actualizacionde 1 segundo, de manera que los datos que se veran en ella seran muy precisos.
Finalmente, nos encontramos con el bloque encargado de cerrar la conexion con el cliente y nos dejara un mensaje en la pantalla diciendonos que el cliente se ha desconectado.

```
void loop() {

  // Change screen every 7.5 seconds (timerDelay variable)
  if ((millis() - lastTimer) > timerDelay) {
    updateScreen();
    Serial.println(displayScreenNum);
    if(displayScreenNum < displayScreenNumMax) {
      displayScreenNum++;
    }
    else {
      displayScreenNum = 0;
    }
    lastTimer = millis();
  }

  //WEB
  WiFiClient client = server.available();   // Escucha a los clientes entrantes

  if (client) {                             // Si se conecta un nuevo cliente
    Serial.println("New Client.");          //
    String currentLine = "";                //
    while (client.connected()) {            // loop mientras el cliente est谩 conectado
      if (client.available()) {             // si hay bytes para leer desde el cliente
        char c = client.read();             // lee un byte
        Serial.write(c);                    // imprime ese byte en el monitor serial
        header += c;
        if (c == '\n') {                    // si el byte es un caracter de salto de linea
          // si la nueva linea est谩 en blanco significa que es el fin del
          // HTTP request del cliente, entonces respondemos:
          if (currentLine.length() == 0) {
            client.println("HTTP/1.1 200 OK");
            client.println("Content-type:text/html");
            client.println("Connection: close");
            client.println();

            // Muestra la p谩gina web
            float temperature = bme.readTemperature();
            float humidity = bme.readHumidity();
            float pressure = bme.readPressure()/100.0F;
            //int ldrReading = map(analogRead(ldr), 0, 4095, 100, 0);
            client.println(paginaInicio + Temperatura + String(temperature) + Humedad + String(humidity) + Presion + String(pressure

            // la respuesta HTTP temina con una linea en blanco
            client.println();
            break;
          } else { // si tenemos una nueva linea limpiamos currentLine
            currentLine = "";
          }
        } else if (c != '\r') {  // si C es distinto al caracter de retorno de carro
          currentLine += c;      // lo agrega al final de currentLine
        }
      }
    }
    // Limpiamos la variable header
    header = "";
    // Cerramos la conexion
    client.stop();
    Serial.println("Client disconnected.");
    Serial.println("");
  }
}
```

Imagenes del display + web:

En las siguientes imagenes mostramos lo que sale en cada una de las pantallas del display y en la web: □□□□□

Aqui vemos una imagen de lo que se ve cuando el programa funciona y nos da una IP en la terminal: □