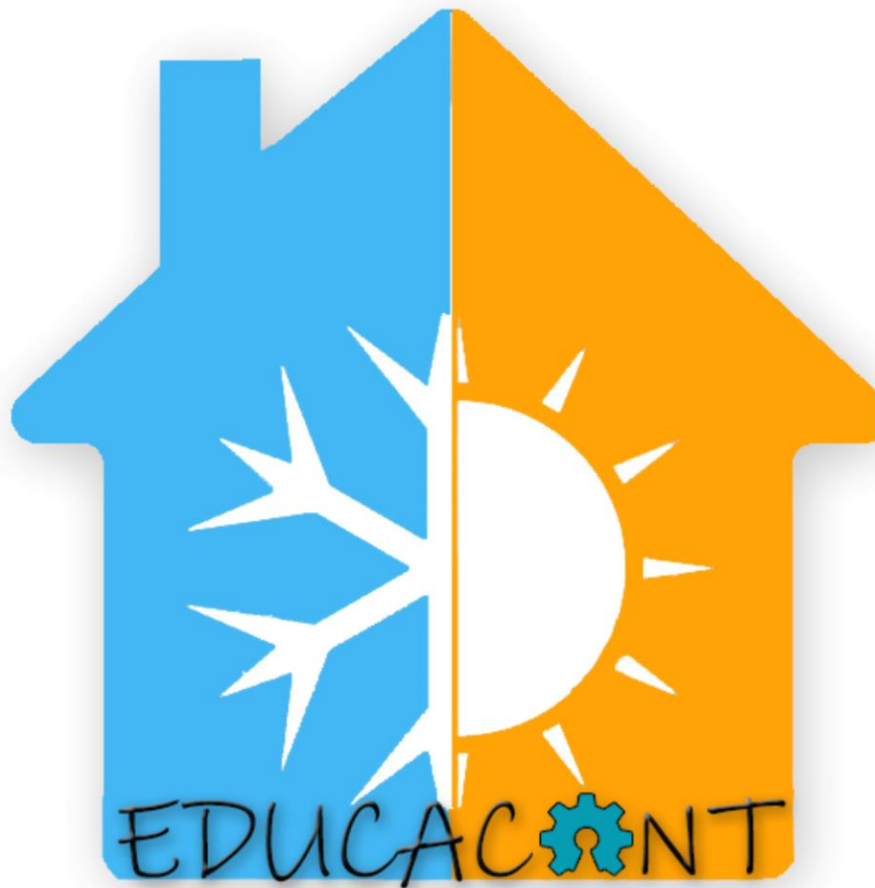


2018/2019

DOCUMENTACIÓN PROYECTO EDUCACONT



Enrique Rodríguez Vela

Contenido

1. DESCRIPCION DEL PROYECTO	2
1.2 Placa Principal del proyecto	2
1.3 Sensores Utilizados	2
1.3.1 Sensor de humedad y temperatura DHT11	2
1.3.2 Sensor Calidad Aire MQ135	3
1.3.3 Sensor de Monóxido de Carbono MQ7	3
1.3 Alimentación	4
1.4 Accesorios.....	4
1.4.1 Pantalla LCD	4
1.4.2 Conexionado del proyecto	5
2.0 Planos eléctricos del proyecto	6
2.1 Esquema Real	6
2.2 Conexiones	6
3.0 Planos Mecánicos.....	7
1.3 Programación	9
1.3.1 Sketch Numero 1: ProgramaFinalEducaCont.ino	9
1.3.2 Programa: ProyectoEducaContLocal.ino.....	15

1. DESCRIPCION DEL PROYECTO

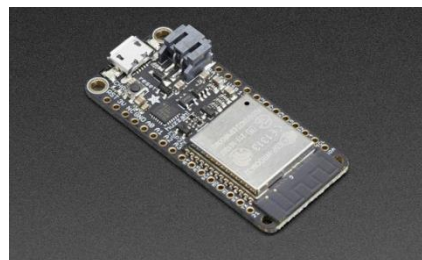
El Proyecto EducaCont, es una estación meteorológica conectada, el objetivo del proyecto es medir la calidad del aire y compartir los datos de forma que el acceso sea libre tanto como para los participantes como cualquier otra persona ajena a este proyecto.

1.2 Placa Principal del proyecto

La placa principal del proyecto es Adafruit Feather – ESP32 que es una plataforma compatible con Arduino creada por Adafruit que pretende ser pequeña, ligera, adecuada para proyectos que funcionen con baterías y fácil de programar añadiendo funcionalidades específicas a cada placa.

Las placas Adafruit Feather incluyen un microcontrolador ATmega32U4 con USB nativo, un cargador de baterías LiPo con conector para programación y recarga y permiten añadir todo tipo de expansiones llamadas "Feather Wings".

El Adafruit Feather HUZZAH32 es una placa con conectividad Wifi basada en el conocido ESP32 (WROOM32). Esta placa no tiene un microcontrolador AVR, sino que utiliza el ESP32 como procesador principal. Es totalmente compatible y programable con Arduino, pero con la gran ventaja que es mucho más potente y corre a 240 MHz gracias a su procesador Tensilica LX6. Además, también tiene un cargador de baterías LiPo integrado, lo que permite utilizarlo con baterías y lo hace ideal para proyectos que funcionen sin cables. Por supuesto es totalmente compatible con las Wings (extensiones) hechas para Feather para ampliar aún más sus funcionalidades.



1.3 Sensores Utilizados

1.3.1 Sensor de humedad y temperatura DHT11

El DHT11 es el sensor encargado de medir la temperatura y la humedad en el ambiente, el sensor lo puedes montar directamente en una placa de prototipaje necesitas poner una resistencia entre datos y VCC, pero también venden DHT11 en la placa PCB ya preparado que es el utilizado en el proyecto.

Los pines del DHT11 son: GND: conexión con tierra, DATA: transmisión de datos, VCC: alimentación



1.3.2 Sensor Calidad Aire MQ135

Este sensor de control de calidad de aire es usado para la detección de contaminación en el medio ambiente, por lo general es implementado en circuitos de control como alarmas en las casas, sitios donde se desea prevenir altos niveles de contaminación a nivel aeróbico como industrias que manejan compuestos químicos que pueden ser nocivos para la salud, especialmente en equipos controladores de calidad de aire en edificios/oficinas.

Este sensor se encarga de la detección de concentración de gas en diversos porcentajes, tal y como los son sus análogos MQ-3/4/5. La señal de salida que proporciona el MQ-135 es dual, de carácter analógico y digital. Respecto a la señal analógica proporcionada, esta viene a ser directamente proporcional al incremento de voltaje. La señal digital, esta presenta niveles TTL por lo que esta señal puede ser procesada por un microcontrolador.



1.3.3 Sensor de Monóxido de Carbono MQ7

Este es un sensor fácil de usar para detección de Monóxido de Carbono (CO), ideal para detectar concentraciones de CO en el aire. El MQ-7 puede detectar concentraciones en el rango de 20 a 2000ppm. El módulo posee una salida analógica que proviene del divisor de voltaje que forma el sensor y una resistencia de carga. También tiene una salida digital que se calibra con un potenciómetro, esta salida tiene un led indicador.



1.3 Alimentación

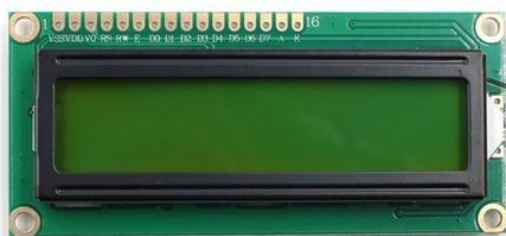
- El proyecto para que funcione de manera autónoma se puede alimentar con una Power Bank (como la que está en la imagen), con la que obtienes los 5 voltios para poder alimentar perfectamente el microcontrolador principal del proyecto, la batería tiene una capacidad de 2200 mAh, que con la pantalla LCD encendida no da para unas 6 horas de autonomía subiendo datos a Thingspeak cada 5 minutos aproximadamente.
- Si el proyecto lo vas a tener en un sitio en el que puedas tener electricidad, puedes conectarlo a un cargador de móvil normal, el cargador debe tener 5 voltios y 1,55 amperios para que pueda funcionar perfectamente



1.4 Accesorios

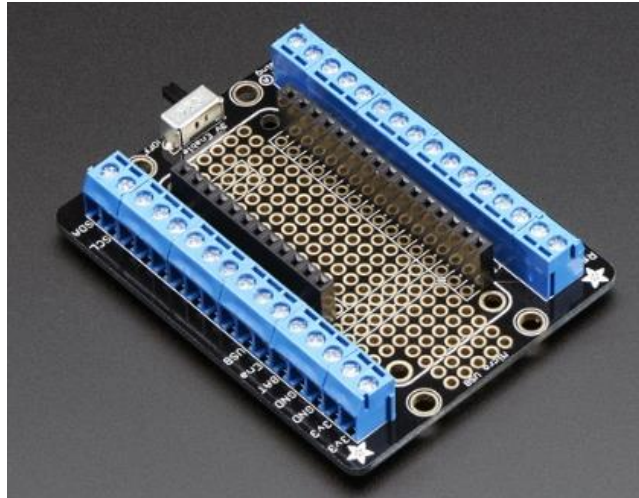
1.4.1 Pantalla LCD

El proyecto consta de una pequeña pantalla LCD que muestra los datos recibidos de los sensores por pantalla, si el proyecto lo tienes de forma local puedes apagar o encender la pantalla desde un botón que esta hecho en una página html, si el proyecto lo que quieres es subir los datos a thingspeak la pantalla se va refrescando en cada subida y muestra los datos de los sensores, la pantalla LCD va conectada por el bus i2c lo que significa que solo tienes que usar 2 cables para enviar los datos a la pantalla llamados SCA y SCL, la alimentación de la LCD va por medio del pin USB de la placa que saca 5 voltios directamente no como los sensores que se alimentan a 5 voltios



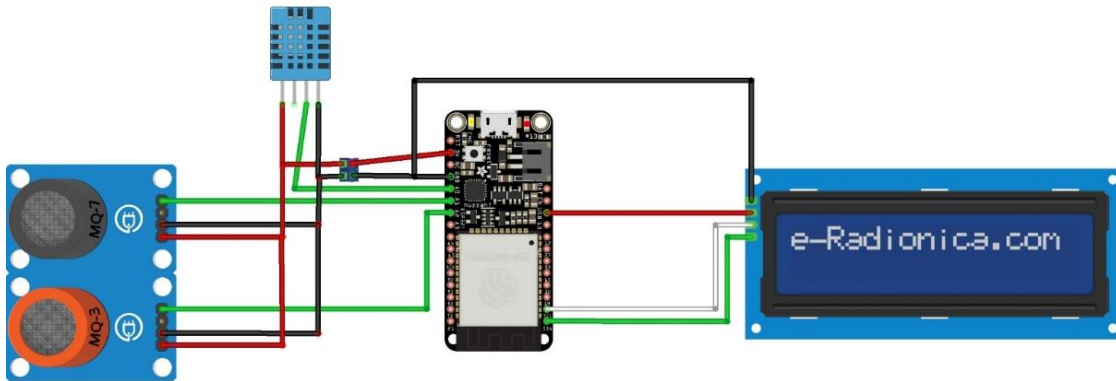
1.4.2 Conexionado del proyecto

Para realizar las conexiones del proyecto se ha utilizado una placa llamada Adafruit Terminal Block Breakout FeatherWing, es una placa diseñada por el mismo distribuidor encargado de diseñar la placa principal del proyecto, esta placa esta diseñada para poder realizar las conexiones de forma más fácil al tener unas bornas de conexión situadas en los extremos de la placa y pequeño interruptor situado en uno de los bordes de la placa para poder encender o apagar los periféricos situados en la placa, la placa tiene unos conectores que vienen ya soldados para que pongas el ESP32 en la placa directamente sin tener que soldarlo.



2.0 Planos eléctricos del proyecto

2.1 Esquema Real



2.2 Conexiones

Adafruit Huzzah ESP32 (Pines)	Conexiones
Pin 3 Voltios	Conectado a una clema para que los reparta entre todos los sensores
Pin GND (Tierra)	Conectado a una clema para que los reparta entre todos los sensores
PIN A0 (Analógico 0)	Conectado al D0 (Digital 0) Sensor DHT11
PIN A1 (Analógico 1)	Conectado al D0 (Digital 0) Sensor MQ7
PIN A2 (Analógico 2)	Conectado al D0 (Digital 0) Sensor MQ135
Pin GND 2 (Tierra)	Conectado al GND de la pantalla LCD
Pin USB	Conectado al VCC de la LCD para que saque 5 Voltios
Pin SCL	Pin SCL
PIN SDA	Pin SDA

- Nota 1: los pines SCL y SDA, son para que se pueda realizar el protocolo i2c con la pantalla LCD
- Nota 2 el proyecto se alimenta tal y como se ha descrito en el apartado 1.3 por una batería conectada al USB de la placa

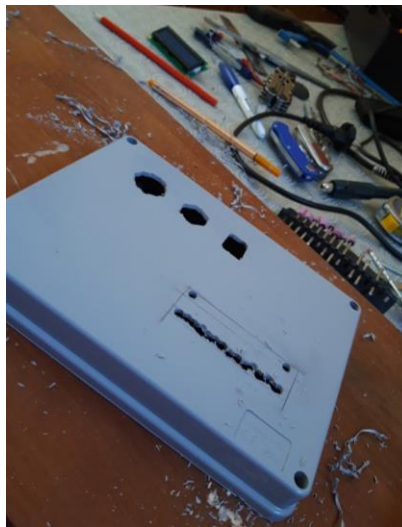
3.0 Planos Mecánicos

Para que quede todo el proyecto ordenado en una caja, se ha diseñado una caja en el programa FreeCAD para que la puedas imprimir con tu impresora 3D. La caja finalmente para el primer prototipo se ha usado una caja para conexiones eléctricas llamada caja de paso.

Se realizaron varios pasos para hacer la caja del prototipo.

1. Se midieron los agujeros por donde tienen salir los diferentes sensores
2. Con una taladradora se hicieron pequeños agujeros para poder ir haciendolos mas grandes para los sensores
3. Con una lima se dio forma finalmente a todo

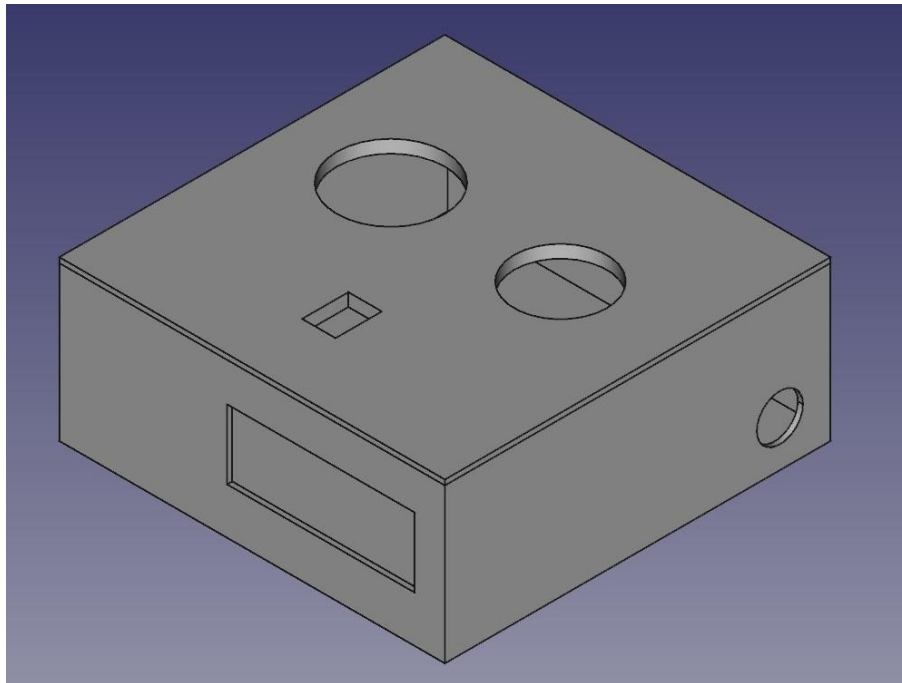
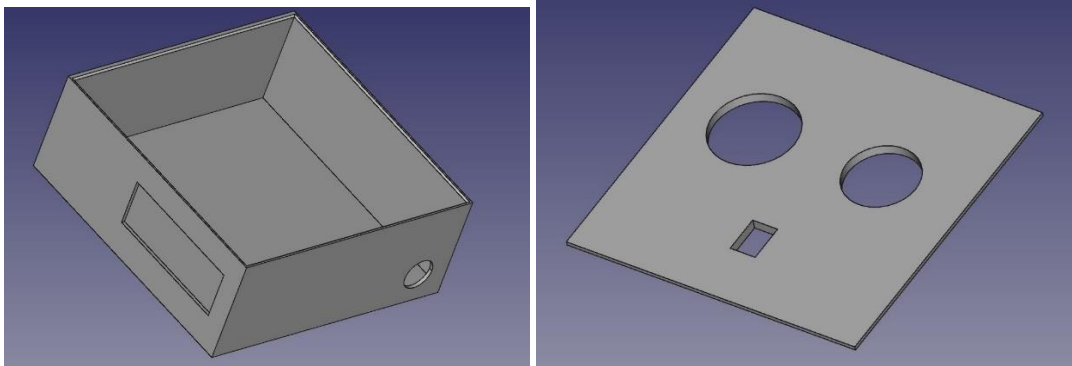
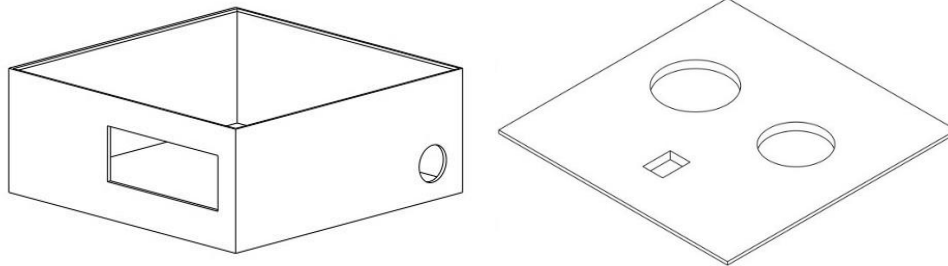
En la siguiente imagen se ve una pequeña parte del proceso de la caja.



En la siguiente secuencia de imágenes vemos como ha quedado finalmente la caja prototipo del proyecto.



Tal y como se ha explicado antes también se ha diseñado la caja en freecad un programa para hacer diseños 3D en las siguientes imágenes se ve cómo quedaría la tapa de la caja y la base de la caja.



1.3 Programación

El proyecto está programado en el IDE de Arduino para poder utilizar esta placa con este IDE, para poder instalarlo hay que seguir estos pasos:

1. Descargue e instale el último Arduino IDE Windows Installer de arduino.cc
2. Descargue e instale Git desde git-scm.com
3. Inicie la interfaz gráfica de usuario de Git y ejecute los siguientes pasos:
4. Seleccionar Clonar Repositorio Existente
5. Seleccionar origen y destino
6. Ubicación de la fuente: <https://github.com/espressif/arduino-esp32.git>
7. Directorio de destino:
 - a. C:/Users/[Tu_Nombre_de_Usuario]/Documents/Arduino/hardware/espressif/esp32
8. Cambie esto a su ubicación de Sketchbook si tiene un directorio diferente que aparece debajo de la “Ubicación de Sketchbook” en las preferencias de Arduino.
9. Haga clic en Clonar para iniciar la clonación del repositorio
 - a. Abrir
C:/Users/[Tu_Nombre_de_Usuario]/Documents/Arduino/hardware/espressif/esp32/tools y doble click en get.exe
10. Cuando get.exe termine, debería ver los siguientes archivos en el directorio
11. Enchufe su tarjeta ESP32 y espere a que los controladores instalen (o instale manualmente los que sean necesarios)

Una vez seguidos estos pasos ya lo tienes todo listo para poder cargar los programas que se usan en este proyecto.

1.3.1 Sketch Numero 1: ProgramaFinalEducaCont.ino

Este programa lo que hace es mandar datos a una página llamada Thingspeak.io, ¿Por qué thingspeak? Porque ThingSpeak te ofrece la posibilidad de ir guardando diferentes tipos de datos, cada cual organizado en un canal distinto. Para crear este canal tendremos que rellenar un formulario donde como máximo podremos registrar hasta 8 campos diferentes.

ThingSpeak™ Channels Apps Community Support

Percentage complete 70%

Channel ID 446518

Name Proyecto Educacont

Description Estacion de temperaturas y gases, hecha con un ESP32, el sensor DHT11, MQ7 y MQ135

Field 1 Temperatura ☒

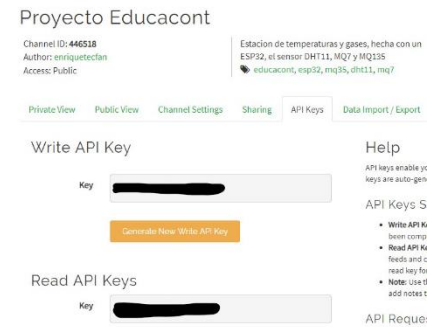
Field 2 Humedad ☒

Field 3 Lectura MQ7 ☒

Field 4 Lectura MQ135 ☒

Una vez hemos completado la configuración y guardados los cambios, necesitaremos apuntar un par de datos que nos proporciona ThingSpeak y que deberemos incluir en el código:

- El número de nuestro canal (Channel ID)
- La clave de escritura (Write API Key)



En cuanto a la parte de programación del ESP32, para poder utilizar el DHT 11, el sensor MQ7, el sensor MQ135 y la pantalla LCD para enviar datos a Thingspeak vamos a necesitar tener instalados en nuestro IDE de arduino algunas librerías:

- La librería para el DHT 11. Que e la librería DHT.h
- La librería «ThingSpeak.h»
- La librería de la pantalla es LiquidCrystal_PCF8574.h
- Todas las librerías las descargamos desde Programa – Incluir Librería – Gestionar Librerías todo ello en el IDE de arduino

Después de todo esto también necesitamos la librería WIFI.h que ya viene incluida cuando instalas el ESP32 para poder usar:

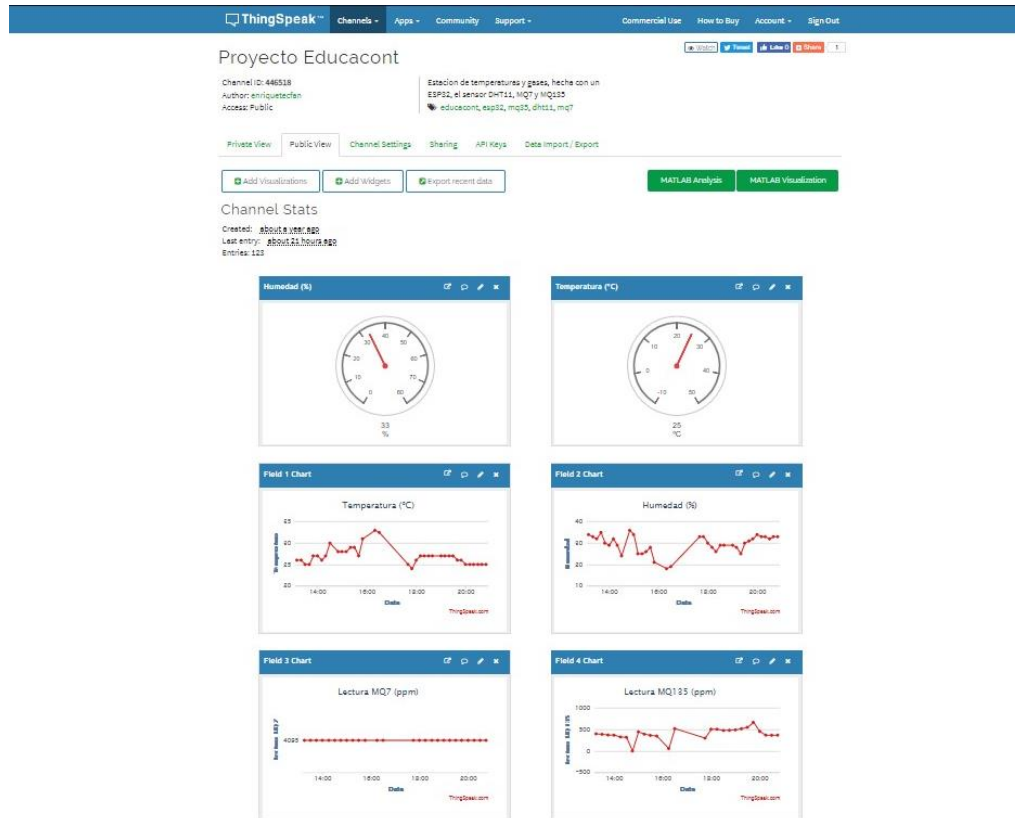
- Nombre de vuestra conexión WiFi
- Contraseña WiFi
- Número de canal de ThingSpeak
- ThingSpeak write API Key

El programa ejemplo lee cada 10 minutos la temperatura, humedad y valores de los sensores de gas, muestra los valores por el puerto serie y también los muestra por la pantalla LCD y los enviar a nuestro canal de ThingSpeak.

```
//-----
//PROGRAMA:ProgramaFinalEducaCont.ino (Adafruit Huzzah ESP-32)
//Autor: Enrique Rodriguez Vela (Junio-2018)
//Descripción: Programa final EducaCont
//-----

#include <WiFi.h>
const char* ssid      = "*****";
const char* password = "*****";
const char* host = "api.thingspeak.com";
const char* THINGSPEAK_API_KEY = "*****";
const boolean IS_METRIC = true;
```

Una vez configurado el canal y enviado datos se verá de la siguiente manera



A continuación, está el programa final para usarlo en el proyecto.

```

1. //-----
2. //PROGRAMA:ProgramaFinalEducaCont.ino (Adafruit Huzzah ESP-32)
3. //Autor: Enrique Rodriguez Vela (Junio-2018)
4. //Descripción: Programa final EducaCont
5. //-----
6.
7. #include <WiFi.h>
8. const char* ssid      = "*****";
9. const char* password = "*****";
10. const char* host = "api.thingspeak.com";
11. const char* THINGSPEAK_API_KEY = "*****";
12. const boolean IS_METRIC = true;
13.
14. // LCD Librerias
15. #include <Wire.h>
16. #include <LiquidCrystal_PCF8574.h>
17. LiquidCrystal_PCF8574 lcd(0x27); // Direccion LCD 0x27
18. int show;
19. //DHT Librerias
20. #include <DHT.h>
21. #define DHTPIN A0
22. #define DHTTYPE DHT11
23. DHT dht(DHTPIN, DHTTYPE);

```

```

24.
25.     // MQ7
26.     #define MQ7 A1
27.     int valor_mq7;
28.
29.     // MQ135
30.     #define MQ135 A2
31.     int valor_mq135;
32.     void setup() {
33.         Serial.begin(115200);
34.         Serial.println();
35.         Serial.print("Connecting to ");
36.         Serial.println(ssid);
37.
38.         WiFi.begin(ssid, password);
39.
40.         while (WiFi.status() != WL_CONNECTED) {
41.             delay(500);
42.             Serial.print(".");
43.         }
44.
45.         Serial.println("");
46.         Serial.println("WiFi connected");
47.         Serial.println("IP address: ");
48.         Serial.println(WiFi.localIP());
49.
50.         int error;
51.         Wire.begin();
52.         Wire.beginTransmission(0x27);
53.         Serial.println("LCD...");
54.         dht.begin();
55.         lcd.begin(16, 2);
56.         lcd.setBacklight(255);
57.
58.         pinMode(MQ7, INPUT);
59.         pinMode(MQ135, INPUT);
60.         pinMode(13, OUTPUT);
61.     }
62.
63.     void loop() {
64.         Serial.print("connecting to ");
65.         Serial.println(host);
66.
67.         // Usamos WifiClient para hacer las conexiones, con el puerto
80
68.         WifiClient client;
69.         const int httpPort = 80;
70.         if (!client.connect(host, httpPort)) {
71.             Serial.println("connection failed");
72.             return;
73.         }
74.         // Esperamos para relaizar la medidas
75.         delay(2000);
76.
77.         // Leemos los valores de los sensores
78.         //Serial DHT11

```

```

79.     float h = dht.readHumidity();
80.     float t = dht.readTemperature();
81.     Serial.println("Temperatura: ");
82.     Serial.println(t);
83.     Serial.println("Humedad: ");
84.     Serial.println(h);
85.
86.     //Serial MQ7
87.     valor_mq7 = analogRead(MQ7);
88.     Serial.println("Valor MQ7: ");
89.     Serial.print(valor_mq7);
90.     Serial.print(" ppm");
91.     Serial.println(" ");
92.
93.     //Serial MQ135
94.     valor_mq135 = analogRead(MQ135);
95.     Serial.println("Valor MQ135: ");
96.     Serial.print(valor_mq135);
97.     Serial.print(" ppm");
98.     Serial.println(" ");
99.
100.    Serial.println("-----");
101.    delay(5000);
102.
103.    //Comenzamos a poner los valores en el lcd
104.    //1° Humedad con el DHT11
105.    lcd.clear();
106.    lcd.setBacklight(255);
107.    lcd.setCursor(4, 0);
108.    lcd.print("Humedad");
109.    lcd.setCursor(5, 1);
110.    lcd.print(h);
111.    lcd.print("%");
112.
113.    delay(5000);
114.
115.    //2° Lectura Sensor MQ7
116.    lcd.clear();
117.    lcd.setBacklight(255);
118.    lcd.setCursor(4, 0);
119.    lcd.print("Lec. MQ7:");
120.    lcd.setCursor(4, 1);
121.    lcd.print(valor_mq7);
122.    lcd.print(" ppm");
123.
124.    //3° Lectura Sensor MQ135
125.    lcd.clear();
126.    lcd.setBacklight(255);
127.    lcd.setCursor(4, 0);
128.    lcd.print("Lec.MQ135:");
129.    lcd.setCursor(4, 1);
130.    lcd.print(valor_mq135);
131.    lcd.print(" ppm");
132.
133.    //4° Temperatura con el DHT11
134.    lcd.clear();

```

```

135.     lcd.setBacklight(255);
136.     lcd.setCursor(2, 0);
137.     lcd.print("Temperatura");
138.     lcd.setCursor(5, 1);
139.     lcd.print(t);
140.     lcd.print((char)223);
141.
142.
143.     delay(5000);
144.
145.     // Creamos la URL de respuesta en TingSpeak
146.     String url = "/update?api_key=";
147.     url += THINGSPEAK_API_KEY;
148.     url += "&field1=";
149.     url += String(t);
150.     url += "&field2=";
151.     url += String(h);
152.     url += "&field3=";
153.     url += String(valor_mq7);
154.     url += "&field4=";
155.     url += String(valor_mq135);
156.
157.     Serial.print("Requesting URL: ");
158.     Serial.println(url);
159.
160.     // Esto enviara la respuesta al servidor
161.     client.print(String("GET ") + url + " HTTP/1.1\r\n" +
162.                  "Host: " + host + "\r\n" +
163.                  "Connection: close\r\n\r\n");
164.     delay(10);
165.     while (!client.available()) {
166.         delay(100);
167.         Serial.print(".");
168.     }
169.     while (client.available()) {
170.         String line = client.readStringUntil('\r');
171.         Serial.print(line);
172.     }
173.
174.     // Añadimos un retraso para limitar el número de escrituras en
    Thinspeak
175.     int duracionDelay = 600; //En segundos
176.     for (int i = 0; i < duracionDelay; i++) { //Esto es debido a
        que el máximo que el Arduino puede procesar con precisión es 5000ms o 5
        segundos
177.         delay (1000);
178.     }
179.
180. } // End Programa
181. [/code]

```

1.3.2 Programa: ProyectoEducaContLocal.ino

Este es el segundo programa del proyecto lo que hace es mandar datos a una página web HTML que genera el, también en la página web he diseñado un botón que lo que hace es encender o apagar la pantalla LCD.

En cuanto a la parte de programación del ESP32, para poder utilizar el DHT 11, el sensor MQ7, el sensor MQ135 y la pantalla LCD necesitamos algunas librerías:

- La librería para el DHT 11. Que es la librería DHT.h
- La librería «ThingSpeak.h»
- La librería de la pantalla es LiquidCrystal_PCF8574.h
- Todas las librerías las descargamos desde Programa – Incluir Librería – Gestionar Librerías todo ello en el IDE de arduino

Después de todo esto también necesitamos la librería WiFi.h que ya viene incluida cuando instalas el ESP32 para poder usar:

- Nombre de vuestra conexión WiFi
- Contraseña WiFi

El programa ejemplo lee cada 10 minutos la temperatura, humedad y valores de los sensores de gas, muestra los valores por el puerto serie y también los muestra por la pantalla LCD y por la página web html de forma local sin guardar los valores.

```

1. //-----
2. //PROGRAMA: ProyectoEducaContLocal.ino (Adafruit Huzzah ESP-32)
3. //Autor: Enrique Rodriguez Vela (Mayo - 2018)
4. //Descripción: Programa final EducaCont de forma local
5. //-----
6.
7. // Load Wi-Fi library
8. #include <WiFi.h>
9.
10.    // Replace with your network credentials
11.    const char* ssid      = "*****";
12.    const char* password = "*****";
13.
14.    // Set web server port number to 80
15.    WiFiServer server(80);
16.
17.    // Variable to store the HTTP request
18.    String header;
19.    //DHT
20.    #include <DHT.h>
21.    #define DHTPIN A0
22.    #define DHTTYPE DHT11
23.    DHT dht(DHTPIN, DHTTYPE);
24.
25.    // MQ7

```



```

26.  #define MQ7 A2
27.  int valor_mq7;
28.
29.  // MQ135
30.  #define MQ135 A3
31.  int valor_mq135;
32.
33.  // LCD Librerias
34.  #include <Wire.h>
35.  #include <LiquidCrystal_PCF8574.h>
36.  LiquidCrystal_PCF8574 lcd(0x27); // Direccion LCD 0x27
37.  int show;
38.
39.  String lcdstate = "off";
40.
41.  void setup() {
42.      Serial.begin(115200);
43.      Serial.print("Connecting to ");
44.      Serial.println(ssid);
45.      WiFi.begin(ssid, password);
46.      while (WiFi.status() != WL_CONNECTED)
47.      {
48.          delay(500);
49.          Serial.print(".");
50.      }
51.      Serial.println("");
52.      Serial.println("WiFi connected.");
53.      Serial.println("IP address: ");
54.      Serial.println(WiFi.localIP());
55.      server.begin();
56.
57.      pinMode(DHTPIN, INPUT);
58.      pinMode(MQ7, INPUT);
59.      pinMode(MQ135, INPUT);
60.      pinMode(13, OUTPUT);
61.
62.      Wire.begin();
63.      Wire.beginTransmission(0x27);
64.      Serial.println("LCD...");
65.      dht.begin();
66.      lcd.begin(16, 2);
67.      lcd.setBacklight(0);
68.
69.      lcd.clear();
70.      lcd.setBacklight(255);
71.      lcd.setCursor(0, 0);
72.      lcd.print("HECHO POR:");
73.      lcd.setCursor(0, 1);
74.      lcd.print("KIKE RODRIGUEZ");
75.      delay(2000);
76.
77.      lcd.clear();
78.      lcd.setBacklight(255);
79.      lcd.setCursor(0, 0);
80.      lcd.print("IP SERVIDOR:");
81.      lcd.setCursor(0, 1);

```

```

82.     lcd.print(WiFi.localIP());
83. }
84.
85. void loop() {
86.
87.     //Valores Sensores
88.     float h = dht.readHumidity();
89.     float t = dht.readTemperature();
90.     valor_mq7 = analogRead(MQ7);
91.     valor_mq135 = analogRead(MQ135);
92.
93.     WiFiClient client = server.available(); // Listen for
incoming clients
94.
95.     if (client) { // If a new client
connects,
96.         Serial.println("New Client."); // print a message
out in the serial port
97.         String currentLine = ""; // make a String to
hold incoming data from the client
98.         while (client.connected()) { // loop while the
client's connected
99.             if (client.available()) { // if there's bytes
to read from the client,
100.                 char c = client.read(); // read a byte, then
101.                 Serial.write(c); // print it out the
serial monitor
102.                 header += c;
103.                 if (c == '\n') { // if the byte is a
newline character
104.                     // if the current line is blank, you got two newline
characters in a row.
105.                     // that's the end of the client HTTP request, so send a
response:
106.                     if (currentLine.length() == 0) {
107.                         client.println("HTTP/1.1 200 OK");
108.                         client.println("Content-type:text/html");
109.                         client.println("Connection: close");
110.                         client.println();
111.
112.                         // turns the GPIOs on and off
113.                         if (header.indexOf("GET /lcd/on") >= 0) {
114.                             Serial.println("LCD ON");
115.                             lcdstate = "on";
116.                             //Comenzamos a poner los valores en el lcd
117.                             //1º Humedad con el DHT11
118.                             lcd.clear();
119.                             lcd.setBacklight(255);
120.                             lcd.setCursor(4, 0);
121.                             lcd.print("Humedad");
122.                             lcd.setCursor(5, 1);
123.                             lcd.print(h);
124.                             lcd.print("%");
125.
126.                             delay(1000);
127.

```

```

128.          //2° Lectura Sensor MQ7
129.          lcd.clear();
130.          lcd.setBacklight(255);
131.          lcd.setCursor(4, 0);
132.          lcd.print("Lec. MQ7:");
133.          lcd.setCursor(4, 1);
134.          lcd.print(valor_mq7);
135.          lcd.print(" ppm");
136.
137.          delay(1000);
138.
139.          //3° Lectura Sensor MQ7
140.          lcd.clear();
141.          lcd.setBacklight(255);
142.          lcd.setCursor(4, 0);
143.          lcd.print("Lec.MQ135:");
144.          lcd.setCursor(4, 1);
145.          lcd.print(valor_mq135);
146.          lcd.print(" ppm");
147.
148.          delay(1000);
149.
150.          //4° Temperatura con el DHT11
151.          lcd.clear();
152.          lcd.setBacklight(255);
153.          lcd.setCursor(2, 0);
154.          lcd.print("Temperatura");
155.          lcd.setCursor(5, 1);
156.          lcd.print(t);
157.          lcd.print((char)223);
158.
159.          delay(1000);
160.      }
161.      if (header.indexOf("GET /lcd/off") >= 0) {
162.          Serial.println("LCD OFF");
163.          lcdstate = "off";
164.          lcd.clear();
165.          lcd.setBacklight(0);
166.      }
167.
168.      client.println("<!DOCTYPE html><html>");
169.      client.println("<head><meta
name=\"viewport\" content=\"width=device-width, initial-scale=1\">");
170.      client.println("<link
rel=\"icon\" href=\"data:,\">>");
171.      client.println("<style>html { font-family: Helvetica;
display: inline-block; margin: 0px auto; text-align: center;});");
172.      client.println(".button { background-color: #4CAF50;
border: none; color: white; padding: 16px 40px;");
173.      client.println("text-decoration: none; font-size:
30px; margin: 2px; cursor: pointer;});");
174.      client.println(".button2 {background-color:
#FF0000;}</style></head>");
175.      client.println("<h1>");
176.      client.println("<center>MEDIDAS SENSORES
EDUCACONT</center>");








```

```

177.         client.println("</h1>");
178.         client.println("<body>");
179.         client.println("<center>Temperatura en la
    habitacion:");
180.         client.println(t);
181.         client.println("<body>&#8451</center>");
182.         client.println(" <body>");
183.         client.println("<center>Humedad en la habitacion:");
184.         client.println(h);
185.         client.println("<body>&#37</center>");
186.         client.println("<body>");
187.         client.println("<center>Valor del Sensor MQ7:");
188.         client.println("<body>");
189.         client.println(valor_mq7);
190.         client.println("ppn</center>");
191.         client.println("<body>");
192.         client.println("<center>Valor del Sensor MQ135:");
193.         client.println("<body>");
194.         client.println(valor_mq135);
195.         client.println("ppn</center>");
196.
197.         if (lcdstate == "off") {
198.             client.println("<center><p><a
    href=\\\"/lcd/on\\\"><button class=\\\"button\\\">Mostrar datos
    LCD</button></a></p></center>");
199.             } else {
200.                 client.println("<center><p><a
    href=\\\"/lcd/off\\\"><button class=\\\"button button2\\\">Apagar
    LCD</button></a></p></center>");
201.             }
202.             client.println("</body></html>");
203.
204.             // The HTTP response ends with another blank line
205.             client.println();
206.             // Break out of the while loop
207.             break;
208.         } else { // if you got a newline, then clear
    currentLine
209.             currentLine = "";
210.         }
211.         } else if (c != '\\r') { // if you got anything else but
    a carriage return character,
212.             currentLine += c; // add it to the end of the
    currentLine
213.         }
214.     }
215. }
216. // Clear the header variable
217. header = "";
218. // Close the connection
219. client.stop();
220. Serial.println("Client disconnected.");
221. Serial.println("");
222. }
223. }

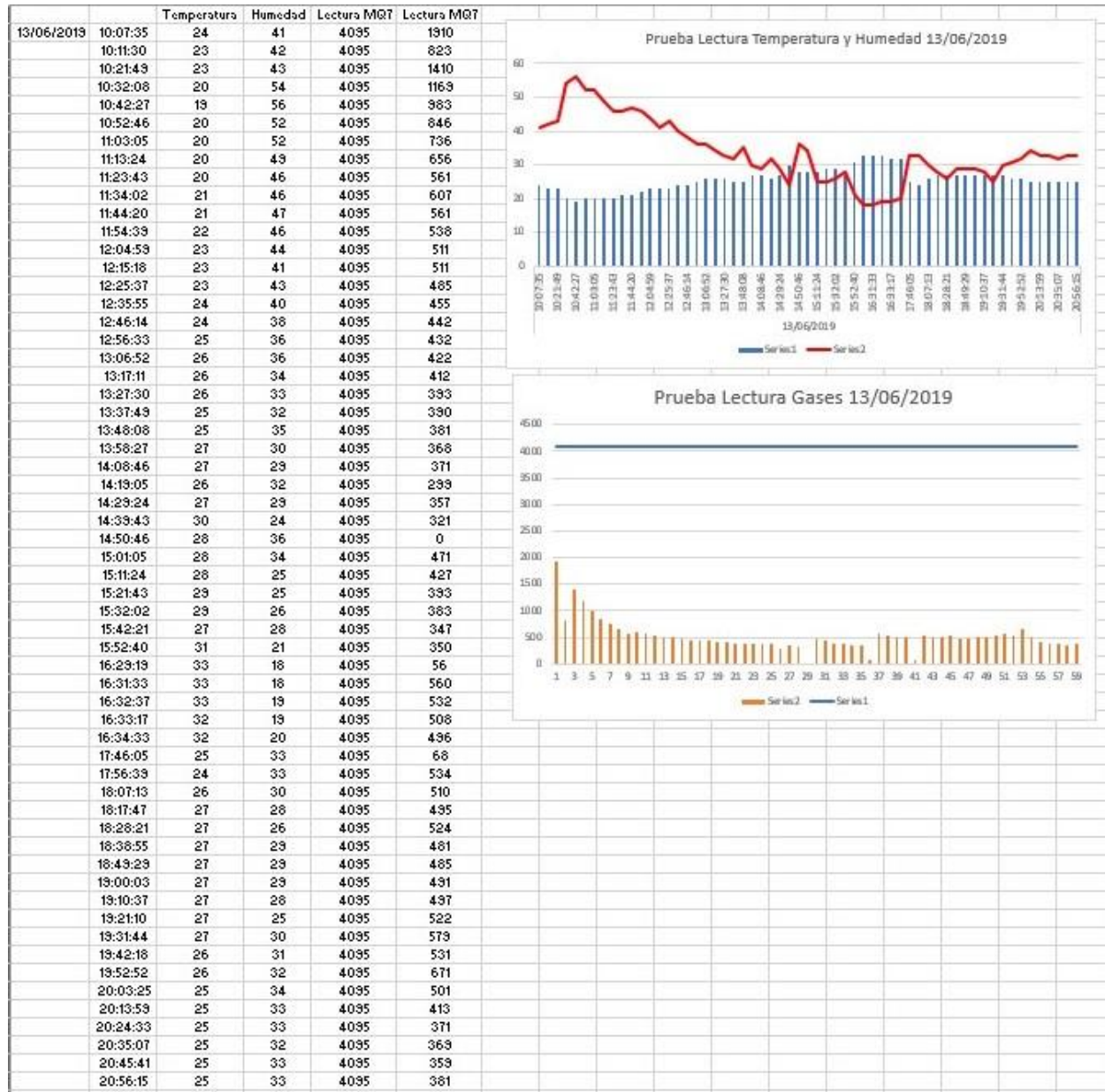
```

1.4 Presupuesto del proyecto

Artículo	Cantidad	Referencia	Coste	Foto
Adafruit Huzzah ESP-32	1	ESP32	21,90 €	
Sensor Temperatura DHT11	1	DHT11	2.10 €	
Sesnor de Gas MQ7	1	MQ7	3.49	
Sensor de Gas MQ135	1	MQ135	6,26 €	
Placa de conexiones	1		15.90 €	
Cables	10		0.50 €	
Caja de Paso	1		5.50 €	
Coste total del proyecto			27.49	

1.5 Pruebas de funcionamiento

En la siguiente prueba se ha usado una batería de 2600mAh que empezó a las: 10:12 y acabo a las 15:50 con pantalla LCD encendida y mandando datos cada 10 min, luego se usó otra batería 600mAh Empieza a las: 17:46 y acaba a las: 20:56 con pantalla LCD apagada encendiéndose cuando manda datos cada 10 min.



1.6 Enlaces Importantes

Para poder acceder a los archivos del proyecto esta todo publicado en mi GitHub, desde hay os podéis bajar todos los archivos si deseáis hacer el proyecto.

Pagina de GitHub: <https://github.com/enriquetecfan11/Proyecto-EducaCont>