****

**Informe Practica Profesional Supervisada**

**Alumno:**

Kevin Roges

DNI: 38.175.596

Mail: [kevin\_roges@yahoo.com.ar](mailto:kevin_roges@yahoo.com.ar)

**Carrera**

Ingeniería Mecánica con Orientación a Mecatrónica

**Índice**

Observaciones...................................................................................2

1. Presentación..................................................................................3

1.1Introducción.........................................................................3

1.2 Objetivos.............................................................................4

1.3 Alcance...............................................................................5

1.4 Resumen............................................................................6

2. Ingeniería de la solución................................................................7

2.1 Especificaciones técnicas...................................................7

2.2 Detalle de pines a utilizar...................................................7

2.3 Materiales a utilizar.............................................................7

2.3.1 Presupuesto...........................................................7

2.3.2 Criterio de selección de cada material...................8

2.4 Especificaciones de Diseño................................................8

2.5 Software a utilizar...............................................................8

2.5.1 Funcionamiento del proyecto.................................9

2.5.2 Código del programa……….................................11

2.6 Explicación del funcionamiento del código.......................20

2.7 Diagramas de conexionado..............................................23

3. Conclusión...................................................................................24**Observaciones**

**1. Presentación**

* 1. **Introducción**

Se realizará un secador de filamento utilizado en impresión 3D, apto para plásticos como el PLA, PETG, TPU, ABS, PA, PPT, entre otros.

El mismo consta de un cajón armado en melamina con capacidad para ocho carretes de filamento.

En el interior encontraremos una resistencia plana de 300 mm x 300 mm de 500 W y dos ventiladores recirculadores que facilitaran el proceso de convección

Todo esto va a estar comandado por un ESP32 y además tendremos un panel con cuatro botones y un display LED.

En este display podremos visualizar un sistema de menú con cuatro pantallas. La principal, donde podremos ver la humedad, temperatura y su grafica en tiempo, temperatura y tiempo objetivos. En la segunda pantalla vamos a poder setear temperatura y tiempo objetivo. En la tercera se puede controlar la resistencia y los ventiladores de forma manual. Y en la ultima podremos visualizar la dirección IP y el estado de conexión a la red WiFi.

Aprovechando, que el ESP32, tiene conexión WiFi se armó un web Server, con la idea de poder controlar el secador de manera remota

* 1. **Objetivos**
* Con este proyecto se busca obtener un producto de mayor capacidad en comparación a las ofertas del mercado. Ya que actualmente los secadores de filamento que encontramos son para uno o dos rollos.
* Con la implementación de la conexión wifi se busca poder centralizar el manejo en un solo dispositivo. Ya que muchas de las impresoras también son controladas de manera remota, poder visualizar y detener el proceso de secado también es algo fundamental
* Este sistema podría tener la facilidad de adaptarse a varías necesidades, ya que el proceso de control no varía. Es cuestión de adaptarlo al volumen necesario y potencia requerida
  1. **Alcance**
* El control de la resistencia será mediante relé, tipo ON-OFF. Se descarta el control PID, ya que la inercia térmica de la resistencia no es muy elevada
* Teniendo en cuenta, que el cajón será de madera, el interior se revestirá con cartón mineral para evitar sobrecalentamiento sobre el enchapado y así no produzca vapores.
* El control del dispositivo sea tanto desde el panel como de manera remota
  1. **Resumen**

Como se mencionó antes, se fabricará un secador de filamento con capacidad para ocho rollos.

Con este se busca mejorar la capacidad de secado en comparación con los que podemos encontrar en el mercado.

Uno de los grandes problemas que hay en la impresión 3D y a veces puede que se le de poca importancia, es la humedad. Hay algunos plásticos que son más higroscópicos que otros y esto mas tarde nos trae problemas en la impresión, como mala adherencia de capa o la formación de “pelillos” en la pieza impresa. Ocasionando perdidas de calidad, perdida de material y tiempo de producción.

A esto le sumamos que muchas veces en granjas de impresión, son muchos los rollos expuestos a la intemperie. Ocasionando demoras en secado de filamento, ya que un plástico como el PLA, el mas utilizado, puede demorar aproximadamente 5 hs en secar

El funcionamiento en bastante sencillo, se setea la temperatura de secado (específica para cada tipo de plástico) y un tiempo. Al llegar este a cero se detiene el equipo

1. **– Ingeniería de la solución**
   1. **Especificaciones técnicas**

* Capacidad para ocho rollos de filamento
* Potencia de resistencia 500 W
* Temperatura máxima de secado 70 °C
* Aislación térmica para el interior de la cabina
* Se debe operar mediante una red wifi a partir de una pagina con una interfaz amigable con el usuario, la cual sea capaz de indicar las mediciones de temperatura objetivo y actual, tiempo objetivo y humedad, además de un botón para detener el equipo
* La conexión inalámbrica debe ser estable y segura para garantizar la comunicación fluida entre microcontrolador y el aplicativo
* La estructura del secador debe ser robusta y bien aislada para garantizar una temperatura de secado estable
* Debe ser fácil de operar, tanto para la configuración del seteo como la manipulación de los filamentos
  1. **Detalles de pines a utilizar**

|  |  |
| --- | --- |
| Cantidad de pines | Elementos |
| 3 | Sensor DHT22 |
| 4 | Pantalla OLED |
| 4 | 4 x pulsadores |
| 2 | Rele estado sólido |
| 2 | Modulo Relay |

Total, de pines: 15

* 1. **Materiales para utilizar**
     1. **Presupuesto**

|  |  |
| --- | --- |
| Artículo | Precio $ |
| Recortes melamina espesor 18 mm | $24.500 |
| Resistencia 500 W | Disponible |
| Display OLED | $7500 |
| Modulo relé | $6000 |
| ESP32 | $9500 |
| Sensor DHT22 | $5800 |
| Fuente conmutada 220vca – 5vcc | $6500 |
| Placa virgen PCB | Disponible |
| Impresiones 3D (soportes, tablero, comando) material PLA, ABS, PETG | $11500 |
| Relé de estado sólido Fotek | $24800 |
| Fuente 12v | Disponible |
| 2x Fan blower 12vcc 5015 | $12000 |

Total $ 108.100

Total USS 90

* + 1. **Criterio de selección de cada material**
* ESP32: Por su tamaño compacto, además de poseer ya integrado el modulo WiFi, necesario para este proyecto
* Placas melamina: Material resistente y económico, en comparación a placas metálicas. Además de su facilidad de trabajar con este
* Piezas plásticas en ABS para el interior, ya que soportan temperaturas de hasta 80 °C continuos
* Tablero eléctrico hecho en PETG, recomendado para aplicaciones eléctricas
  1. **Especificaciones de diseño**

Se construirá con placas de melamina y piezas impresas en 3D de diseño propio.

Como la temperatura máxima de secado será de 70 °C, se optó por la melamina ya que soporta hasta 90 °C. De todas maneras, el interior será recubierto en placas de cartón mineral que funcionaran de aislante.

Las piezas impresas en 3D también se seleccionaron según sus propiedades térmicas y mecánicas. Teniendo en cuenta, que los soportes de los caños, donde van a estar sujetándose las bobinas de filamento de 1kg cada una, van a estar soportando el peso en condiciones de temperaturas de 65 °C como máximo.

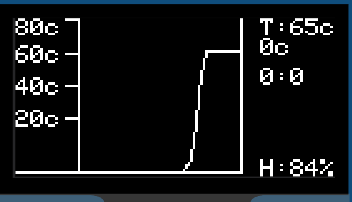
El control de la resistencia es mediante on-off, se descarta control PID ya que complejizaría el código. Además, la inercia térmica es muy baja, teniendo menos de 1 °C de oscilación

* 1. **Software por utilizar**

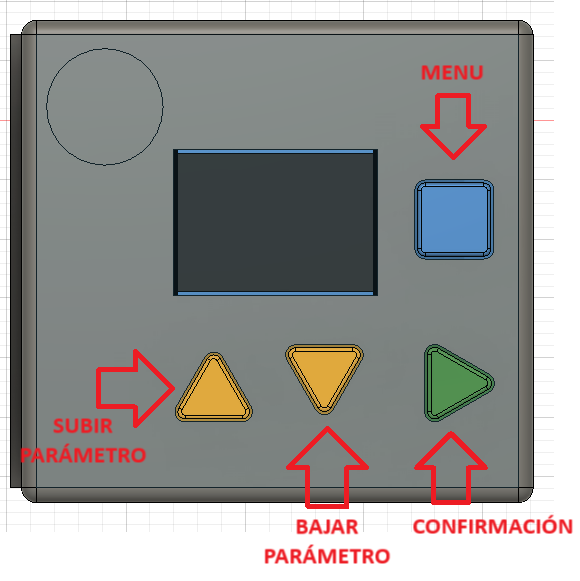
Se programo en lenguaje C desde el Arduino IDE el ESP32. Se implemento un web server alojado en el mismo y una combinación de lenguajes HTML y CSS para el diseño de la página web, cuyo diseño es bien sencillo y solo debe mostrar: Temperatura objetivo, temperatura actual, tiempo de secado y humedad en %. Además del botón para detener el secador

**2.5.1 Funcionamiento del proyecto**

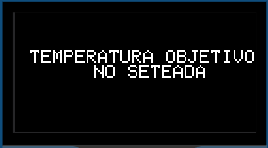
Inicia el ESP32 y lo primero que veremos es la pantalla principal, donde nos indica la temperatura objetivo, temperatura actual, tiempo de seteo y humedad. Junto a un gráfico de tiempo vs temperatura. Esto más que nada para ver si en algún momento el sensor de temperatura marco alguna diferencia. Ya sea por una fuga térmica de la cabina o falla misma del sensor



El comando de control se ve de la siguiente manera



Con la tecla “Menú” alternamos entre las distintas pantallas, “Subir parámetro” y “Bajar parámetro”, como su nombre lo indica, permite aumentar o disminuir los valores de tiempo y temperatura objetivos, como también alternar entre encendido y apagado la resistencia y ventiladores recirculadores (veremos más adelante). El botón “Confirmación” es para confirmar los valores y también iniciar el secado. Este tiene una protección mediante software de no arrancar si no hay una temperatura y tiempo objetivo seteado y despliega el siguiente mensaje en pantalla:



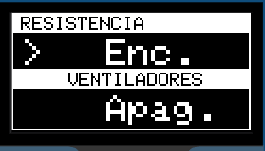
Al presionar el botón “menú” pasamos a la segunda pantalla donde podemos setear la temperatura y tiempo objetivo. Con “subir parámetro” aumentamos de a 5 °C hasta llegar a la temperatura deseada, por ejemplo 55 °C, ideal para secar un plástico como el PLA. Con el botón “confirmación” aceptamos este valor y ahora nos permite ajustar el tiempo objetivo. Este valor lo podemos aumentar o disminuir, con los botones correspondientes, de a 30 minutos.

En caso de querer modificar la temperatura objetivo, presionando nuevamente “confirmación” nos deja modificar este bajando o subiendo



Si presionamos “menú” nuevamente pasamos a la tercer pantalla, donde podemos encender la resistencia y los ventiladores recirculadores de manera manual. Con el botón “subir parámetro” se enciende y el botón “bajar parámetro” se apaga. Al presionar “confirmación” alternamos entre la resistencia y los ventiladores.

Este menú se hizo más que nada para verificar el correcto funcionamiento de estos dos elementos clave del secador, no se recomienda dejar la resistencia encendida de manera manual, ya que la temperatura puede elevarse demasiado



Por último, al presionar “menú” accedemos a la última pantalla, donde solo visualizaremos el estado de conexión a la red wifi y la dirección IP. Esta dirección IP, al ingresarla en un dispositivo, nos permite ingresar al menú de control remoto. Como se mencionó antes veremos el estado de secado y los valores medidos en el interior de la cabina

**2.5.2 Código del programa**

1. //Incluimos las librerias
2. #include "DHTesp.h"
3. #include <Wire.h>
4. #include <Adafruit\_GFX.h>
5. #include <Adafruit\_SSD1306.h>
6. #include <WiFi.h>
7. #include <WebServer.h> // Librería para el servidor web
9. //Iniciamos la pantalla OLED
10. #define ANCHO 128
11. #define ALTO 64
12. #define OLED\_RESET 4
13. Adafruit\_SSD1306 display(ANCHO, ALTO);
14. //Iniciamos sensor DHT22
15. int pinDHT = 15;
16. DHTesp dht;
17. int anteriorMillis = 0;
18. int tiempo = 0;
19. float temperatura = 0;
20. float graficaTemperatura = 0;
21. int x[128]; // Buffer de la gráfica
22. int y[128]; // Buffer secundario de la gráfica
23. float bufferTemperatura[5]; // Buffer para el promedio móvil (5 lecturas)
24. int indexBuffer = 0; // Índice del buffer para el promedio móvil
25. int totalLecturas = 0; // Total de lecturas acumuladas
26. unsigned long intervalo = 1000; // Intervalo en milisegundos para actualizar la gráfica (ajústalo según necesites)
27. unsigned long ultimaActualizacion = 0;
28. int temp\_obj = 0; //Variable de temperatura objetivo
29. int HS\_obj = 0; //Seteable
30. int MIN\_obj = 0; //Seteable
31. int HS\_total = 0; //Mostrar en pantalla
32. int MIN\_total = 0; //Mostrar pantalla
33. int tiempo\_total; //Contempla el HS\_obj+MIN\_obj en segundos
34. int tiempo\_obj = 0; //Variable de tiempo objetivo
35. int menu = 0;
36. int menu2 = 0;
37. int menu3 = 0;
38. int prog = 0; //Flag para programa
39. String valor; //Estado de resistencia
40. String valor2; //Estado de fan
41. const int menu\_ok = 4; //Botón para entrar a menu o volver
42. const int menu\_up = 5; // Botón para aumentar valores
43. const int menu\_down = 2; //Botón para decrecer valores
44. const int menu\_config = 17; //Botón para alternar entre temp y tiempo
45. const int fan = 19; //Salida para encender ventiladores
46. const int resistencia = 18; //Salida para encender resistencia
47. unsigned long tiempo\_restante;
48. unsigned long tiempo\_transcurrido;
49. unsigned long tiempoInicio;
50. bool temporizadorActivo = false;
51. bool confirmTemporizador = false;
52. //Configuración wifi
53. const char\* ssid = "TeleCentro-0923";
54. const char\* password = "Motorola123";
55. // Web server en el puerto 80
56. WebServer server(80);
57. // Función para obtener la página web
58. String getWebPage() {
59. dht.setup(pinDHT, DHTesp::DHT22);
60. TempAndHumidity data = dht.getTempAndHumidity();
61. String html = "<html><head><meta http-equiv='refresh' content='5'/>"
62. "<style>table {font-family: Arial, sans-serif; border-collapse: collapse; width: 100%;}"
63. "td, th {border: 1px solid #dddddd; text-align: left; padding: 8px;}"
64. "tr:nth-child(even) {background-color: #dddddd;}</style></head><body>"
65. "<h2>Estado del Sistema</h2><table><tr><th>Mediciones</th><th>Seteos</th></tr>";
67. html += "<tr><td>Temperatura actual: " + String(data.temperature, 1) + " C</td>";
68. html += "<td>Temperatura objetivo: " + String(temp\_obj) + " C</td></tr>";
70. html += "<tr><td>Humedad actual: " + String(data.humidity, 1) + "%</td>";
71. html += "<td>Temporizador: " + String(HS\_total) + " hs " + String(MIN\_total) + " min</td></tr>";
73. html += "</table><br><form action='/detener' method='POST'><button type='submit'>Detener Secado</button></form>";
75. html += "</body></html>";
77. return html;
78. }
79. // Función para manejar las peticiones de la página principal
80. void handleRoot() {
81. server.send(200, "text/html", getWebPage());
82. }
83. // Función para manejar la acción de detener el secado
84. void handleDetener() {
85. //secadoActivo = false;  // Detiene el secado
86. digitalWrite(18, LOW); // Apaga la resistencia
87. digitalWrite(19, LOW); // Apaga el ventilador
88. server.send(200, "text/html", "<html><body><h2>Secado detenido</h2><a href='/'>Volver</a></body></html>");
89. }
90. void setup() {
91. WiFi.begin(ssid, password);
92. while (WiFi.status() != WL\_CONNECTED) {
93. delay(1000);
94. }
95. Serial.begin(115200);
96. Serial.println(WiFi.localIP());
97. // Inicia el servidor
98. server.on("/", handleRoot);
99. server.on("/detener", HTTP\_POST, handleDetener);
100. server.begin();
101. pinMode(5, INPUT\_PULLDOWN);
102. pinMode(4, INPUT\_PULLDOWN);
103. pinMode(2, INPUT\_PULLDOWN);
104. pinMode(17, INPUT\_PULLDOWN);
105. pinMode(18, OUTPUT);
106. pinMode(19, OUTPUT);
107. temp\_obj=0;
108. Wire.begin();
109. display.begin(SSD1306\_SWITCHCAPVCC, 0x3C);
110. //Inicializamos el dht
111. dht.setup(pinDHT, DHTesp::DHT22);
112. // Inicialización del buffer x[] con valores fuera del rango de la pantalla
113. for(int i=0; i<128; i++){
114. x[i] = 62;  // Inicializa con un valor correspondiente a la línea base (0 °C)
115. }
116. // Inicializa el buffer del promedio móvil
117. for (int i = 0; i < 5; i++) {
118. bufferTemperatura[i] = 0;
119. }
120. }
121. void loop() {
122. if(digitalRead(resistencia)==HIGH){ //Lectura del estado de resistencia
123. valor="Enc.";
124. }else{
125. valor="Apag.";
126. }
127. if(digitalRead(fan)==HIGH){
128. valor2="Enc.";
129. }else{
130. valor2="Apag.";
131. }
132. //Menu
133. if(digitalRead(menu\_ok) == HIGH){// Cambia el estado del menú
134. menu++;
135. menu2=0;
136. menu3=0;
137. delay(200);
138. if(menu==4){
139. menu=0;
140. }
141. }
142. if(menu==0){ //PANTALLA PRINCIPAL CON GRAFICO
143. display.clearDisplay();
144. display.setTextSize(1);
145. display.setTextColor(SSD1306\_WHITE);
146. TempAndHumidity data = dht.getTempAndHumidity();
147. unsigned long actualMillis = millis(); // Tiempo actual en milisegundos
148. // Solo actualiza la gráfica si ha pasado el intervalo deseado
149. if (actualMillis - ultimaActualizacion >= intervalo) {
150. ultimaActualizacion = actualMillis; // Actualiza el tiempo de la última actualización
151. display.clearDisplay(); // Limpia el buffer del display
152. // Dibuja la escala de temperatura
153. display.setCursor(0, 0);
154. display.print(F("80c"));
155. display.setCursor(0, 11);
156. display.print(F("60c"));
157. display.setCursor(0, 24);
158. display.print(F("40c"));
159. display.setCursor(0, 37);
160. display.print(F("20c"));
161. //Dibuja lineas para los numeros de arriba
162. display.drawLine(20, 0, 25, 0, WHITE);
163. display.drawLine(20, 14, 25, 14, WHITE);
164. display.drawLine(20, 27, 25, 27, WHITE);
165. display.drawLine(20, 40, 25, 40, WHITE);
166. // Dibuja eje X y Y
167. display.drawLine(0, 62, 90, 62, WHITE);
168. display.drawLine(25, 62, 25, 0, WHITE);
169. display.drawLine(90, 62, 90, 0, WHITE);
170. // Lee la temperatura del sensor DHT22
171. temperatura = data.temperature;
172. // Actualiza el buffer de promedio móvil
173. bufferTemperatura[indexBuffer] = temperatura;
174. indexBuffer = (indexBuffer + 1) % 5;
175. if (totalLecturas < 5) {
176. totalLecturas++;
177. }
178. // Calcula el promedio móvil
179. float suma = 0;
180. for (int i = 0; i < totalLecturas; i++) {
181. suma += bufferTemperatura[i];
182. }
183. float temperaturaPromedio = suma / totalLecturas;
184. // Escala la temperatura a la posición en la pantalla (0 a 80 °C)
185. graficaTemperatura = map(temperaturaPromedio, 0, 80, 62, 0);
186. // Desplaza el buffer x[] para la nueva lectura
187. for (int i = 0; i < 90; i++) {
188. x[i] = x[i+1];
189. }
190. x[90] = graficaTemperatura; // Asigna el valor escalado al último dato de la matriz
191. // Dibuja líneas para una gráfica más fluida
192. for(int i = 89; i >= 25; i--){
193. display.drawLine(i+1, x[i+1], i, x[i], WHITE); // Conecta puntos adyacentes con una línea
194. }
195. // Imprime la temperatura en texto
196. display.setTextSize(1.8);
197. display.setCursor(98, 0);
198. display.print("T:"+String(data.temperature,0)+"c");
199. display.setCursor(98,57);
200. display.print("H:"+String(data.humidity,0)+"%");
201. display.setCursor(98,8);
202. display.print(String(temp\_obj)+"c");
203. display.setCursor(98,20);
204. display.print(String(HS\_total)+":"+String(MIN\_total));
205. display.display();
206. }
207. //Programa secado
208. if(digitalRead(menu\_config)==HIGH){
209. prog++;
210. delay(200);
211. if(prog==2){
212. prog=0;
213. temporizadorActivo = false;  // Detener temporizador
214. confirmTemporizador = false;  // Resetear confirmación de temporizador
215. digitalWrite(resistencia, LOW);
216. digitalWrite(fan, LOW);
217. }
218. }
219. if(prog==0){
220. tiempoInicio = 0;  // Reiniciar el temporizador
221. tiempo\_transcurrido = 0;
222. tiempo\_restante = 0;
223. temporizadorActivo = false;
224. confirmTemporizador = false;  // Resetear la confirmación del temporizador
225. }
226. if(prog==1){
227. if(temp\_obj==0){
228. display.clearDisplay();
229. display.setCursor(8,20);
230. display.setTextSize(1);
231. display.print("TEMPERATURA OBJETIVO       NO SETEADA");
232. display.display();
233. prog=0;
234. delay(2000);
235. }else{
236. digitalWrite(fan, HIGH);
237. if(data.temperature <= temp\_obj){
238. digitalWrite(resistencia, HIGH);
239. }else{
240. digitalWrite(resistencia, LOW);
241. }
242. if (!temporizadorActivo && !confirmTemporizador && data.temperature >= temp\_obj) {
243. tiempoInicio = millis();  // Inicia el temporizador
244. confirmTemporizador = true;  // Confirma que el temporizador ya inició
245. temporizadorActivo = true;  // Activa el temporizador
246. }
247. if (temporizadorActivo) {
248. tiempo\_total = (HS\_obj \* 3600 + MIN\_obj \* 60) \* 1000;  // Tiempo total en milisegundos
249. tiempo\_transcurrido = millis() - tiempoInicio;
250. tiempo\_restante = tiempo\_total - tiempo\_transcurrido;
251. HS\_total = tiempo\_restante / (3600 \* 1000);  // Horas
252. MIN\_total = (tiempo\_restante % (3600 \* 1000)) / (60 \* 1000);  // Minutos
253. if (HS\_total==0 && MIN\_total==0){
254. digitalWrite(resistencia, LOW);
255. digitalWrite(fan, LOW);
256. prog=0;
257. temporizadorActivo = false;  // Detener temporizador
258. confirmTemporizador = false;  // Resetear confirmación de temporizador
259. display.clearDisplay();
260. display.setCursor(15,25);
261. display.setTextSize(1);
262. display.print("SECADO COMPLETADO");
263. display.display(); //Se despliega el aviso de que termino
264. delay(2000);
265. if(digitalRead(menu\_config)==HIGH){ //Presionar para volver a pantalla principal
266. menu=0;
267. }
268. }
269. }
270. }
271. }
272. }
273. if(menu==1){ //MENU SETEO TEMPERATURA Y TIEMPO
274. if (digitalRead(menu\_config)==HIGH){
275. menu2++;
276. delay(200);
277. display.clearDisplay();
278. if(menu2==2){
279. menu2=0;
280. }
281. }
282. if(menu2==0){
283. display.setCursor(5,14);
284. display.setTextSize(2);
285. display.print(">");
286. if(digitalRead(menu\_up)==HIGH && temp\_obj!=70){
287. temp\_obj += 5;
288. delay(200);
289. }
290. if(digitalRead(menu\_down)==HIGH && temp\_obj!=0){
291. temp\_obj -= 5;
292. delay(200);
293. }
294. }
295. if(menu2==1){
296. display.setCursor(5,44);
297. display.setTextSize(2);
298. display.print(">");
299. if(digitalRead(menu\_up)==HIGH){
300. MIN\_obj += 30;
301. delay(200);
302. }
303. if(digitalRead(menu\_down)==HIGH){
304. MIN\_obj -= 30;
305. delay(200);
306. }
307. if(MIN\_obj >= 60) {
308. HS\_obj += 1;
309. MIN\_obj = 0;
310. }
311. if(MIN\_obj < 0) {
312. HS\_obj -= 1;
313. MIN\_obj = 30;
314. }
315. }
316. display.setTextSize(1);
317. display.setCursor(5,2);
318. display.drawRect(0, 0, 127, 63, WHITE);
319. display.fillRect(0, 0, 127, 11, WHITE);
320. display.fillRect(0, 30, 127, 11, WHITE);
321. display.setTextColor(BLACK, WHITE);
322. display.print("TEMPERATURA OBJETIVO");
323. display.setCursor(30,32);
324. display.setTextColor(BLACK, WHITE);
325. display.print("TIEMPO SECADO");
326. display.setTextColor(WHITE, BLACK);
327. display.setCursor(50,14);
328. display.setTextSize(2);
329. display.print(String(temp\_obj));
330. display.setCursor(80,14);
331. display.print("c");
332. display.setCursor(30,44);
333. display.print(String(HS\_obj)+":"+String(MIN\_obj));
334. display.setCursor(80,44);
335. display.print("hs");
336. display.display();
337. }
338. if(menu==2){//MENU ENC. Y APAG. VENTILADORES Y RESISTENCIA
339. display.clearDisplay();
340. if (digitalRead(menu\_config)==HIGH){
341. menu3++;
342. delay(200);
343. display.clearDisplay();
344. if(menu3==2){
345. menu3=0;
346. }
347. }
348. if(menu3==0){
349. display.setCursor(5,14);
350. display.setTextSize(2);
351. display.print(">");
352. if(digitalRead(menu\_up)==HIGH){
353. valor="Enc.";
354. digitalWrite(resistencia, HIGH);
355. delay(200);
356. }
357. if(digitalRead(menu\_down)==HIGH){
358. valor="Apag.";
359. digitalWrite(resistencia,LOW);
360. delay(200);
361. }
362. }
363. if(menu3==1){
364. display.setCursor(5,44);
365. display.setTextSize(2);
366. display.print(">");
367. if(digitalRead(menu\_up)==HIGH){
368. valor2="Enc.";
369. digitalWrite(fan, HIGH);
370. delay(200);
371. }
372. if(digitalRead(menu\_down)==HIGH){
373. valor2="Apag.";
374. digitalWrite(fan, LOW);
375. delay(200);
376. }
377. }
378. display.setTextSize(1);
379. display.setCursor(32,2);
380. display.drawRect(0, 0, 127, 63, WHITE);
381. display.fillRect(0, 0, 127, 11, WHITE);
382. display.fillRect(0, 30, 127, 11, WHITE);
383. display.setTextColor(BLACK, WHITE);
384. display.print("RESISTENCIA");
385. display.setCursor(30,32);
386. display.setTextColor(BLACK, WHITE);
387. display.print("VENTILADORES");
388. display.setTextColor(WHITE, BLACK);
389. display.setCursor(50,14);
390. display.setTextSize(2);
391. display.print(valor);
392. display.setCursor(50,44);
393. display.print(valor2);
394. display.display();
395. }
396. if(menu==3){
397. display.clearDisplay();
398. display.setTextSize(1);
399. display.setCursor(32,2);
400. display.drawRect(0, 0, 127, 63, WHITE);
401. display.fillRect(0, 0, 127, 11, WHITE);
402. display.fillRect(0, 30, 127, 11, WHITE);
403. display.setTextColor(BLACK, WHITE);
404. display.print("DIRECCION IP");
405. display.setCursor(47,32);
406. display.setTextColor(BLACK, WHITE);
407. display.print("ESTADO");
408. display.setTextColor(WHITE, BLACK);
409. display.setCursor(25,17);
410. display.setTextSize(1);
411. display.print(WiFi.localIP());
412. display.setCursor(40,48);
413. if(WiFi.status() != WL\_CONNECTED){
414. display.print("DESCONECTADO");
415. }else{
416. display.print("CONECTADO");
417. }
418. display.display();
419. }
420. // Atender peticiones web
421. server.handleClient();
422. }

**2.6 Explicación del funcionamiento del código**

**A. Inclusión de Librerías**

El código incluye varias librerías esenciales:

* DHTesp.h: Para manejar el sensor de temperatura y humedad DHT22.
* Wire.h, Adafruit\_GFX.h, Adafruit\_SSD1306.h: Para manejar la pantalla OLED.
* WiFi.h, WebServer.h: Para habilitar el servidor web en el ESP32.

**B. Definiciones e Inicialización**

* **Pantalla OLED**: Se define con Adafruit\_SSD1306 display (ANCHO, ALTO), con un tamaño de 128x64 píxeles.
* **Sensor DHT22**: Se asigna al pin 15 del ESP32.
* **Variables de control**:
  + temp\_obj: Temperatura objetivo.
  + HS\_obj, MIN\_obj: Tiempo de secado configurado por el usuario.
  + HS\_total, MIN\_total: Tiempo restante del secado.
  + valor, valor2: Estados de la resistencia y el ventilador.
  + prog: Estado del programa de secado.
* **Pines de entrada y salida**:
  + Botones (menu\_ok, menu\_up, menu\_down, menu\_config) para navegar en el menú y ajustar valores.
  + Salidas (fan, resistencia) para controlar el ventilador y la resistencia de calentamiento.

**C. Servidor Web**

El ESP32 actúa como un servidor web para visualizar en tiempo real la temperatura, humedad y configuración del secador.

**getWebPage()**

Genera una página HTML con:

* Temperatura y humedad actuales.
* Temperatura objetivo y tiempo restante.
* Un botón para detener el secado.

**handleRoot()**

Maneja las solicitudes a la página principal del servidor.

**handleDetener()**

Apaga la resistencia y el ventilador cuando el usuario detiene el secado desde la interfaz web.

**D. Configuración Inicial (setup())**

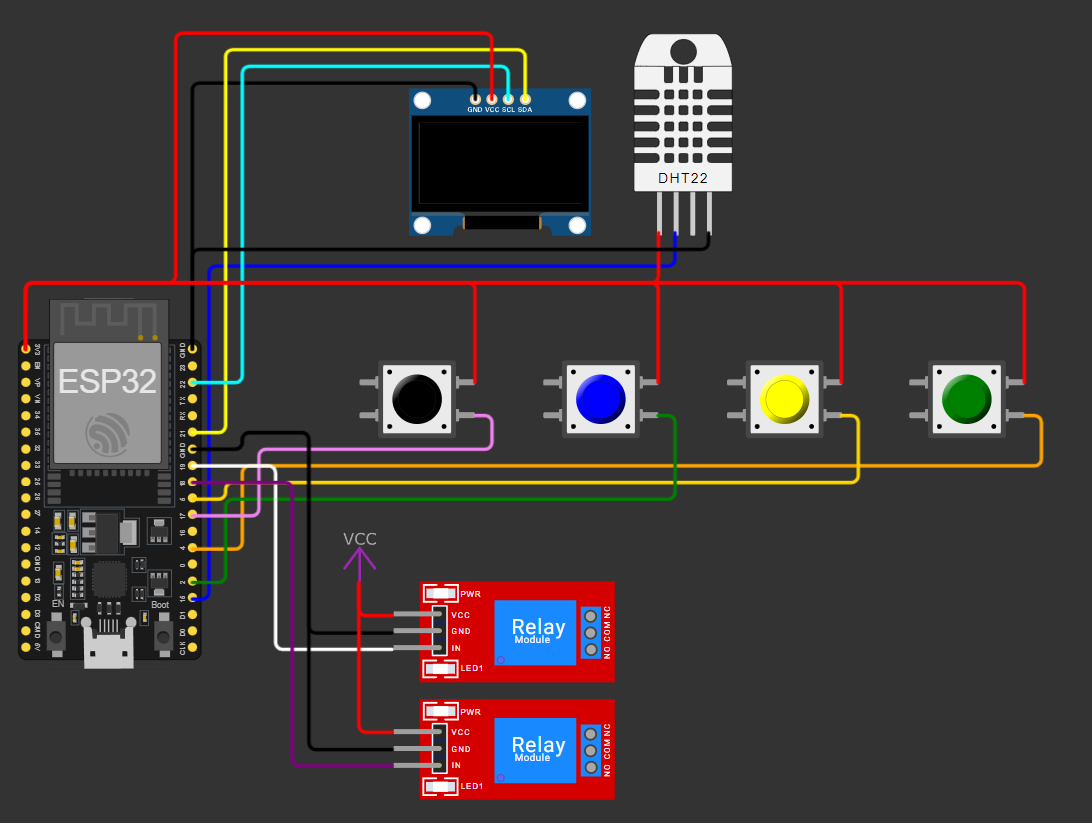
1. **Conexión WiFi** con WiFi.begin(ssid, password).
2. **Configuración del servidor web** en el puerto 80.
3. **Inicialización del sensor DHT22**.
4. **Configuración de pines** de entrada (botones) y salida (resistencia y ventilador).
5. **Inicialización de la pantalla OLED** y buffers de gráficos.

**E. Bucle Principal (loop())**

El código maneja:

1. **Estado de la resistencia y ventilador**.
2. **Menú de configuración**:
   * menu == 0: Muestra la pantalla principal con un gráfico de temperatura en tiempo real.
   * menu == 1: Permite configurar la temperatura y el tiempo de secado.
   * menu == 2: Permite activar o desactivar manualmente el ventilador y la resistencia.
3. **Gráfica de temperatura**:
   * Toma lecturas del sensor DHT22.
   * Aplica un promedio móvil para suavizar la gráfica.
   * Mapea la temperatura al rango de la pantalla OLED.
   * Dibuja una gráfica en tiempo real.
4. **Control del proceso de secado**:
   * Si prog == 1, inicia el secado si la temperatura objetivo está definida.
   * Si la temperatura es menor a la configurada, enciende la resistencia.
   * Una vez alcanzada la temperatura, inicia el temporizador de secado.
   * Cuando el tiempo de secado termina, apaga la resistencia y el ventilador.

**2.7 Diagrama de conexión**



**3 Conclusión**

Se logro el diseño y fabricación del secador de filamento, con capacidad de ocho rollos, utilizando una estructura robusta y un microcontrolador ESP32.

El control de este es bastante amigable a intuitivo, lo que permite una configuración muy rápida. Además, el control remoto por wifi permite al usuario trabajar desde un solo dispositivo sin perder de enfoque el proceso de secado