Informe Practica Profesional Supervisada

Facultad de
INGENIERÍA
Universidad Nacional de Lomas de Zamora

Secadora de Filamento Impresión 3D

Alumno:

Kevin Roges

DNI: 38.175.596

Carrera:

Ingeniería Mecánica con Orientación a Mecatrónica

Índice

Observaciones

1. Presentación

1.1 Introducción

El objetivo de este proyecto fue desarrollar un secador de filamento para impresión 3D, capaz de controlar la temperatura y humedad para mejorar la calidad del material. El diseño se basó en un cajón de melamina con capacidad para ocho carretes de filamento, incorporando un sistema de calefacción y ventilación automatizado mediante un ESP32.

Apto para plásticos como el PLA, PETG, TPU, ABS, PA, PPT, entre otros.

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

Además, se incorporó un panel con cuatro botones y un display LED. En este se puede visualizar un menú con cuatro pantallas. La principal, donde vemos la humedad, temperatura y su grafica en tiempo, temperatura y tiempo objetivos. En la segunda pantalla podemos configurar temperatura y tiempo objetivo. En la tercera se puede controlar la resistencia y los ventiladores de forma manual.

Aprovechando, que el ESP32, tiene conexión WiFi se armó un web Server, con la idea de controlar el secador de manera remota. En la última pantalla del menú, nos indica la dirección IP y el estado de conexión

1.2 Objetivos

- Con este proyecto se buscó 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 podemos 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
- Gracias a su margen de temperatura, se pudo secar gran variedad de filamentos.

1.3 Alcance

- Se opta por un control ON-OFF en lugar de PID debido a su simplicidad, facilidad de implementación y suficiencia para este tipo de sistema. Un control PID podría mejorar la precisión, pero requeriría una calibración más compleja sin un beneficio significativo en este caso.
- Teniendo en cuenta, que el cajón es de madera, el interior se revistio 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

1.4 Resumen

La capacidad de secado, en comparación con los que podemos encontrar en el mercado, se logró aumentar.

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 más 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.

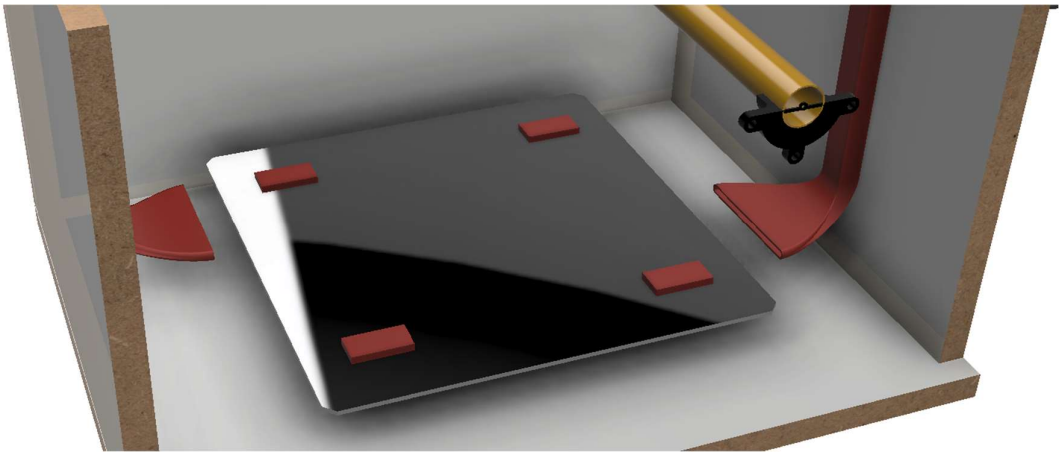
A esto le sumamos que muchas veces en granjas de impresión, son muchos los rollos expuestos a la intemperie. Ocasionando demoras en secado de filamento, ya que un plástico como el PLA, el más utilizado, puede demorar aproximadamente 5 horas en secar

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

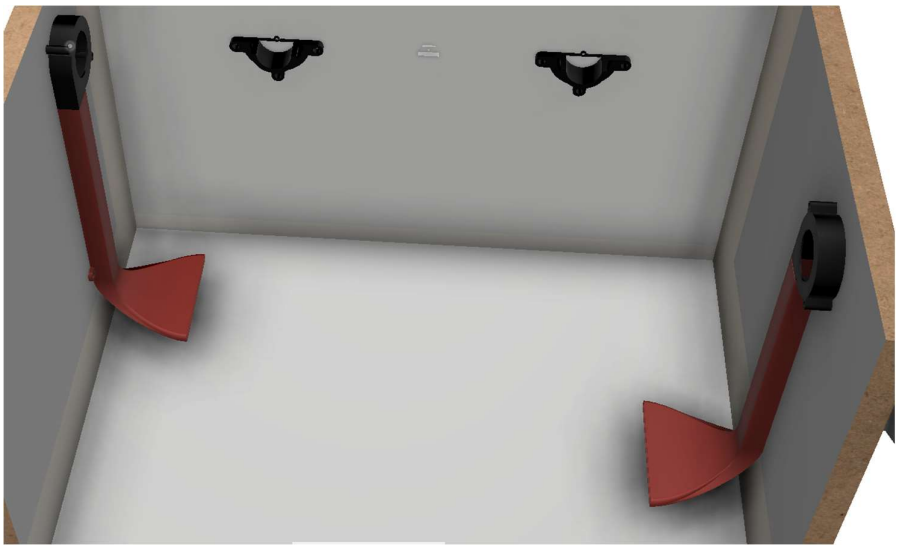
2 – Ingeniería de la solución

2.1 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 página 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 como la manipulación de los filamentos



Resistencia placa 300mm X 300mm



Ventiladores para mejorar la transferencia térmica por convección

2.2 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

2.3 Materiales para utilizar

2.3.1 Presupuesto

Artículo	Precio (Pesos Argentinos)
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

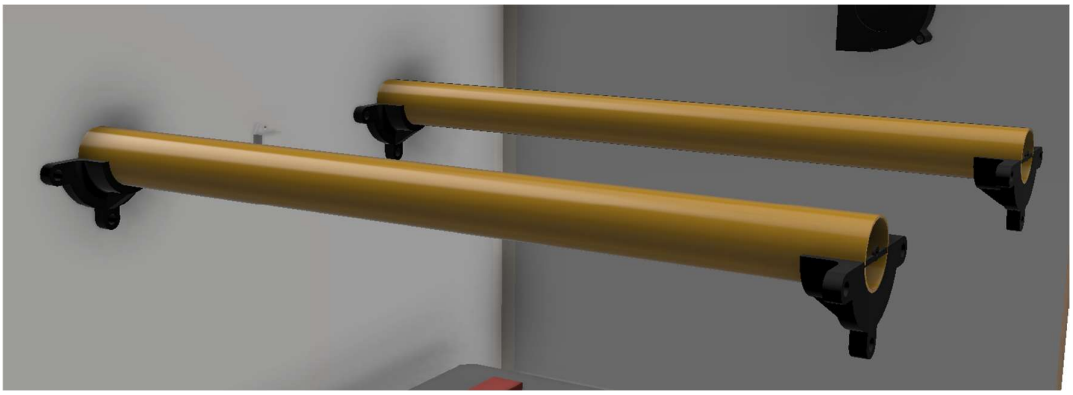
Total $ 108.100

Total, USS 90

2.3.2 Criterio de selección de cada material

- ESP32: Por su tamaño compacto, además de poseer ya integrado el módulo 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

Tablero eléctrico, impreso en material PETG



Soportes de caños, impresos en ABS

2.4 Especificaciones de diseño

Se construyo 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 fue recubierto en placas de cartón mineral que funcionaran de aislante.

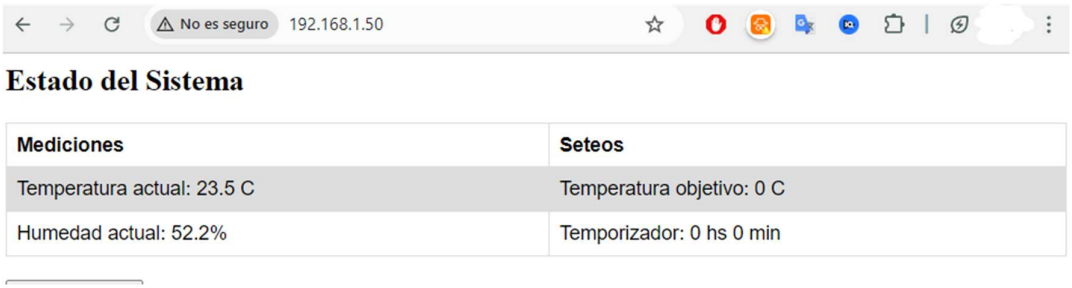
Las piezas impresas en 3D también se seleccionaron según sus propiedades térmicas y mecánicas. Teniendo en cuenta, que los soportes de los caños, donde están sujetas las bobinas de filamento de 1kg cada una, tienen que soportar el peso en condiciones de temperaturas de 70 °C como máximo.

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.

Capacidad de secado hasta ocho bobinas completas

2.5 Software utilizado

El ESP32 se programó en lenguaje C desde el Arduino IDE. 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

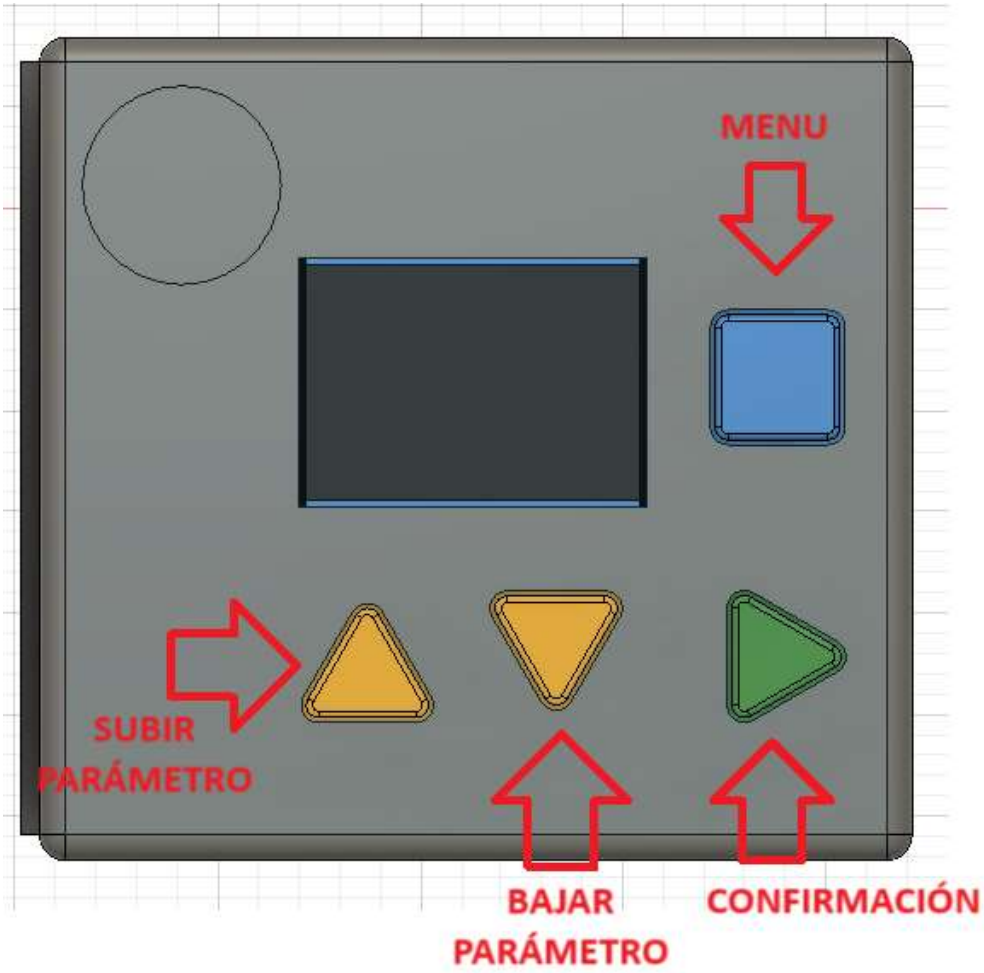


Página principal del web server

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 configurado 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

```
//Incluimos las librerias
#include "DHTesp.h"
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>
#include <WiFi.h>
#include <WebServer.h> // Librería para el servidor web

//Iniciamos la pantalla OLED
#define ANCHO 128
#define ALTO 64
#define OLED_RESET 4
Adafruit_SSD1306 display(ANCHO, ALTO);
//Iniciamos sensor DHT22
int pinDHT = 15;
DHTesp dht;

int anteriorMillis = 0;
int tiempo = 0;
float temperatura = 0;
float graficaTemperatura = 0;

int x[128]; // Buffer de la gráfica
int y[128]; // Buffer secundario de la gráfica
float bufferTemperatura[5]; // Buffer para el promedio móvil (5 lecturas)
int indexBuffer = 0; // Índice del buffer para el promedio móvil
int totalLecturas = 0; // Total de lecturas acumuladas
unsigned long intervalo = 1000; // Intervalo en milisegundos para actualizar la gráfica (ajústalo según necesites)
unsigned long ultimaActualizacion = 0;
int temp_obj = 0; //Variable de temperatura objetivo
int HS_obj = 0; //Seteable
```

```
int MIN_obj = 0; //Seteable
int HS_total = 0; //Mostrar en pantalla
int MIN_total = 0; //Mostrar pantalla
int tiempo_total; //Contempla el HS_obj+MIN_obj en segundos
int tiempo_obj = 0; //Variable de tiempo objetivo
int menu = 0;
int menu2 = 0;
int menu3 = 0;
int prog = 0; //Flag para programa
String valor; //Estado de resistencia
String valor2; //Estado de fan
const int menu_ok = 4; //Botón para entrar a menu o volver
const int menu_up = 5; // Botón para aumentar valores
const int menu_down = 2; //Botón para decrecer valores
const int menu_config = 17; //Botón para alternar entre temp y tiempo
const int fan = 19; //Salida para encender ventiladores
const int resistencia = 18; //Salida para encender resistencia
unsigned long tiempo_restante;
unsigned long tiempo_transcurrido;
unsigned long tiempoInicio;
bool temporizadorActivo = false;
bool confirmTemporizador = false;

//Configuración wifi
const char* ssid = "TeleCentro-0923";
const char* password = "Motorola123";

// Web server en el puerto 80
WebServer server(80);

// Función para obtener la página web
String getWebPage() {
  dht.setup(pinDHT, DHTesp::DHT22);
  TempAndHumidity data = dht.getTempAndHumidity();
  String html = "<html><head><meta http-equiv='refresh' content='5'/>"
                "<style>table {font-family: Arial, sans-serif; border-collapse: collapse; width: 100%;}"
                "td, th {border: 1px solid #dddddd; text-align: left; padding: 8px;}"
                "tr:nth-child(even) {background-color: #dddddd;}</style></head><body>"
                "<h2>Estado del Sistema</h2><table><tr><th>Mediciones</th><th>Seteos</th></tr>";

  html += "<tr><td>Temperatura actual: " + String(data.temperature, 1) + " C</td>";
  html += "<td>Temperatura objetivo: " + String(temp_obj) + " C</td></tr>";

  html += "<tr><td>Humedad actual: " + String(data.humidity, 1) + "%</td>";
  html += "<td>Temporizador: " + String(HS_total) + " hs " + String(MIN_total) + " min</td></tr>";
```

```
79
80   html += "</table><br><form action='/detener' method='POST'><button
    type='submit'>Detener Secado</button></form>";
81
82   html += "</body></html>";
83
84   return html;
85 }
86
87 // Función para manejar las peticiones de la página principal
88 void handleRoot() {
89   server.send(200, "text/html", getWebPage());
90 }
91
92 // Función para manejar la acción de detener el secado
93 void handleDetener() {
94   //secadoActivo = false;  // Detiene el secado
95   digitalWrite(18, LOW); // Apaga la resistencia
96   digitalWrite(19, LOW); // Apaga el ventilador
97   server.send(200, "text/html", "<html><body><h2>Secado detenido</h2><a
    href='/'>Volver</a></body></html>");
98 }
99
100    void setup() {
101      WiFi.begin(ssid, password);
102      while (WiFi.status() != WL_CONNECTED) {
103        delay(1000);
104      }
105      Serial.begin(115200);
106      Serial.println(WiFi.localIP());
107      // Inicia el servidor
108      server.on("/", handleRoot);
109      server.on("/detener", HTTP_POST, handleDetener);
110      server.begin();
111
112      pinMode(5, INPUT_PULLDOWN);
113      pinMode(4, INPUT_PULLDOWN);
114      pinMode(2, INPUT_PULLDOWN);
115      pinMode(17, INPUT_PULLDOWN);
116      pinMode(18, OUTPUT);
117      pinMode(19, OUTPUT);
118      temp_obj=0;
119      Wire.begin();
120      display.begin(SSD1306_SWITCHCAPVCC, 0x3C);
121      //Inicializamos el dht
122      dht.setup(pinDHT, DHTesp::DHT22);
123     // Inicialización del buffer x[] con valores fuera del rango de la pantalla
124      for(int i=0; i<128; i++){
125        x[i] = 62;  // Inicializa con un valor correspondiente a la línea base (0
    °C)
```

```
126     }
127     // Inicializa el buffer del promedio móvil
128     for (int i = 0; i < 5; i++) {
129       bufferTemperatura[i] = 0;
130     }
131   }
132
133   void loop() {
134     if(digitalRead(resistencia)==HIGH){ //Lectura del estado de resistencia
135       valor="Enc.";
136     }else{
137       valor="Apag.";
138     }
139     if(digitalRead(fan)==HIGH){
140       valor2="Enc.";
141     }else{
142       valor2="Apag.";
143     }
144     //Menu
145     if(digitalRead(menu_ok) == HIGH){// Cambia el estado del menú
146       menu++;
147       menu2=0;
148       menu3=0;
149       delay(200);
150       if(menu==4){
151         menu=0;
152       }
153     }
154     if(menu==0){ //PANTALLA PRINCIPAL CON GRAFICO
155       display.clearDisplay();
156       display.setTextSize(1);
157       display.setTextColor(SSD1306_WHITE);
158       TempAndHumidity data = dht.getTempAndHumidity();
159
160       unsigned long actualMillis = millis(); // Tiempo actual en milisegundos
161
162       // Solo actualiza la gráfica si ha pasado el intervalo deseado
163       if (actualMillis - ultimaActualizacion >= intervalo) {
164         ultimaActualizacion = actualMillis; // Actualiza el tiempo de la última
    actualización
165         display.clearDisplay(); // Limpia el buffer del display
166         // Dibuja la escala de temperatura
167         display.setCursor(0, 0);
168         display.print(F("80c"));
169         display.setCursor(0, 11);
170         display.print(F("60c"));
171         display.setCursor(0, 24);
172         display.print(F("40c"));
173         display.setCursor(0, 37);
174         display.print(F("20c"));
```

```
175         //Dibuja lineas para los numeros de arriba
176         display.drawLine(20, 0, 25, 0, WHITE);
177         display.drawLine(20, 14, 25, 14, WHITE);
178         display.drawLine(20, 27, 25, 27, WHITE);
179         display.drawLine(20, 40, 25, 40, WHITE);
180         // Dibuja eje X y Y
181         display.drawLine(0, 62, 90, 62, WHITE);
182         display.drawLine(25, 62, 25, 0, WHITE);
183         display.drawLine(90, 62, 90, 0, WHITE);
184         // Lee la temperatura del sensor DHT22
185         temperatura = data.temperature;
186         // Actualiza el buffer de promedio móvil
187         bufferTemperatura[indexBuffer] = temperatura;
188         indexBuffer = (indexBuffer + 1) % 5;
189         if (totalLecturas < 5) {
190           totalLecturas++;
191         }
192         // Calcula el promedio móvil
193         float suma = 0;
194         for (int i = 0; i < totalLecturas; i++) {
195           suma += bufferTemperatura[i];
196         }
197         float temperaturaPromedio = suma / totalLecturas;
198         // Escala la temperatura a la posición en la pantalla (0 a 80 °C)
199         graficaTemperatura = map(temperaturaPromedio, 0, 80, 62, 0);
200         // Desplaza el buffer x[] para la nueva lectura
201         for (int i = 0; i < 90; i++) {
202           x[i] = x[i+1];
203         }
204         x[90] = graficaTemperatura; // Asigna el valor escalado al último dato
    de la matriz
205         // Dibuja líneas para una gráfica más fluida
206         for(int i = 89; i >= 25; i--){
207           display.drawLine(i+1, x[i+1], i, x[i], WHITE); // Conecta puntos
    adyacentes con una línea
208         }
209         // Imprime la temperatura en texto
210         display.setTextSize(1.8);
211         display.setCursor(98, 0);
212         display.print("T:"+String(data.temperature,0)+"c");
213         display.setCursor(98,57);
214         display.print("H:"+String(data.humidity,0)+"%");
215         display.setCursor(98,8);
216         display.print(String(temp_obj)+"c");
217         display.setCursor(98,20);
218         display.print(String(HS_total)+":"+String(MIN_total));
219         display.display();
220       }
221       //Programa secado
222       if(digitalRead(menu_config)==HIGH){
```

```
223         prog++;
224         delay(200);
225         if(prog==2){
226           prog=0;
227           temporizadorActivo = false;  // Detener temporizador
228           confirmTemporizador = false;  // Resetear confirmación de
    temporizador
229           digitalWrite(resistencia, LOW);
230           digitalWrite(fan, LOW);
231           }
232       }
233       if(prog==0){
234           tiempoInicio = 0;  // Reiniciar el temporizador
235           tiempo_transcurrido = 0;
236           tiempo_restante = 0;
237           temporizadorActivo = false;
238           confirmTemporizador = false;  // Resetear la confirmación del
    temporizador
239
240       }
241       if(prog==1){
242         if(temp_obj==0){
243           display.clearDisplay();
244           display.setCursor(8,20);
245           display.setTextSize(1);
246           display.print("TEMPERATURA OBJETIVO       NO SETEADA");
247           display.display();
248           prog=0;
249           delay(2000);
250           }else{
251             digitalWrite(fan, HIGH);
252             if(data.temperature <= temp_obj){
253               digitalWrite(resistencia, HIGH);
254             }else{
255               digitalWrite(resistencia, LOW);
256             }
257             if (!temporizadorActivo && !confirmTemporizador &&
    data.temperature >= temp_obj) {
258               tiempoInicio = millis();  // Inicia el temporizador
259               confirmTemporizador = true;  // Confirma que el temporizador ya
    inició
260               temporizadorActivo = true;  // Activa el temporizador
261             }
262             if (temporizadorActivo) {
263               tiempo_total = (HS_obj * 3600 + MIN_obj * 60) * 1000;  // Tiempo
    total en milisegundos
264               tiempo_transcurrido = millis() - tiempoInicio;
265               tiempo_restante = tiempo_total - tiempo_transcurrido;
266
267               HS_total = tiempo_restante / (3600 * 1000);  // Horas
```

```
268             MIN_total = (tiempo_restante % (3600 * 1000)) / (60 * 1000);  //
    Minutos
269             if (HS_total==0 && MIN_total==0){
270               digitalWrite(resistencia, LOW);
271               digitalWrite(fan, LOW);
272               prog=0;
273               temporizadorActivo = false;  // Detener temporizador
274               confirmTemporizador = false;  // Resetear confirmación de
    temporizador
275               display.clearDisplay();
276               display.setCursor(15,25);
277               display.setTextSize(1);
278               display.print("SECADO COMPLETADO");
279               display.display(); //Se despliega el aviso de que termino
280               delay(2000);
281               if(digitalRead(menu_config)==HIGH){ //Presionar para volver a
    pantalla principal
282                 menu=0;
283               }
284             }
285           }
286         }
287       }
288     }
289     if(menu==1){ //MENU SETEO TEMPERATURA Y TIEMPO
290       if (digitalRead(menu_config)==HIGH){
291         menu2++;
292         delay(200);
293         display.clearDisplay();
294         if(menu2==2){
295           menu2=0;
296         }
297       }
298       if(menu2==0){
299         display.setCursor(5,14);
300         display.setTextSize(2);
301         display.print(">");
302         if(digitalRead(menu_up)==HIGH && temp_obj!=70){
303           temp_obj += 5;
304           delay(200);
305         }
306         if(digitalRead(menu_down)==HIGH && temp_obj!=0){
307           temp_obj -= 5;
308           delay(200);
309         }
310       }
311       if(menu2==1){
312         display.setCursor(5,44);
313         display.setTextSize(2);
314         display.print(">");
```

```
315         if(digitalRead(menu_up)==HIGH){
316           MIN_obj += 30;
317           delay(200);
318         }
319         if(digitalRead(menu_down)==HIGH){
320           MIN_obj -= 30;
321           delay(200);
322         }
323         if(MIN_obj >= 60) {
324         HS_obj += 1;
325         MIN_obj = 0;
326         }
327         if(MIN_obj < 0) {
328         HS_obj -= 1;
329         MIN_obj = 30;
330         }
331       }
332       display.setTextSize(1);
333       display.setCursor(5,2);
334       display.drawRect(0, 0, 127, 63, WHITE);
335       display.fillRect(0, 0, 127, 11, WHITE);
336       display.fillRect(0, 30, 127, 11, WHITE);
337       display.setTextColor(BLACK, WHITE);
338       display.print("TEMPERATURA OBJETIVO");
339       display.setCursor(30,32);
340       display.setTextColor(BLACK, WHITE);
341       display.print("TIEMPO SECADO");
342       display.setTextColor(WHITE, BLACK);
343       display.setCursor(50,14);
344       display.setTextSize(2);
345       display.print(String(temp_obj));
346       display.setCursor(80,14);
347       display.print("c");
348       display.setCursor(30,44);
349       display.print(String(HS_obj)+":"+String(MIN_obj));
350       display.setCursor(80,44);
351       display.print("hs");
352       display.display();
353     }
354     if(menu==2){//MENU ENC. Y APAG. VENTILADORES Y
    RESISTENCIA
355       display.clearDisplay();
356       if (digitalRead(menu_config)==HIGH){
357         menu3++;
358         delay(200);
359         display.clearDisplay();
360         if(menu3==2){
361           menu3=0;
362         }
363       }
```

```
364       if(menu3==0){
365         display.setCursor(5,14);
366         display.setTextSize(2);
367         display.print(">");
368         if(digitalRead(menu_up)==HIGH){
369           valor="Enc.";
370           digitalWrite(resistencia, HIGH);
371           delay(200);
372         }
373         if(digitalRead(menu_down)==HIGH){
374           valor="Apag.";
375           digitalWrite(resistencia,LOW);
376           delay(200);
377         }
378       }
379       if(menu3==1){
380         display.setCursor(5,44);
381         display.setTextSize(2);
382         display.print(">");
383         if(digitalRead(menu_up)==HIGH){
384           valor2="Enc.";
385           digitalWrite(fan, HIGH);
386           delay(200);
387         }
388         if(digitalRead(menu_down)==HIGH){
389           valor2="Apag.";
390           digitalWrite(fan, LOW);
391           delay(200);
392         }
393       }
394       display.setTextSize(1);
395       display.setCursor(32,2);
396       display.drawRect(0, 0, 127, 63, WHITE);
397       display.fillRect(0, 0, 127, 11, WHITE);
398       display.fillRect(0, 30, 127, 11, WHITE);
399       display.setTextColor(BLACK, WHITE);
400       display.print("RESISTENCIA");
401       display.setCursor(30,32);
402       display.setTextColor(BLACK, WHITE);
403       display.print("VENTILADORES");
404       display.setTextColor(WHITE, BLACK);
405       display.setCursor(50,14);
406       display.setTextSize(2);
407       display.print(valor);
408       display.setCursor(50,44);
409       display.print(valor2);
410       display.display();
411     }
412     if(menu==3){
413       display.clearDisplay();
```

```
414       display.setTextSize(1);
415       display.setCursor(32,2);
416       display.drawRect(0, 0, 127, 63, WHITE);
417       display.fillRect(0, 0, 127, 11, WHITE);
418       display.fillRect(0, 30, 127, 11, WHITE);
419       display.setTextColor(BLACK, WHITE);
420       display.print("DIRECCION IP");
421       display.setCursor(47,32);
422       display.setTextColor(BLACK, WHITE);
423       display.print("ESTADO");
424       display.setTextColor(WHITE, BLACK);
425       display.setCursor(25,17);
426       display.setTextSize(1);
427       display.print(WiFi.localIP());
428       display.setCursor(40,48);
429       if(WiFi.status() != WL_CONNECTED){
430         display.print("DESCONECTADO");
431       }else{
432         display.print("CONECTADO");
433       }
434       display.display();
435     }
436     // Atender peticiones web
437     server.handleClient();
438   }
```

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:

I. **Estado de la resistencia y ventilador**.

II. **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.

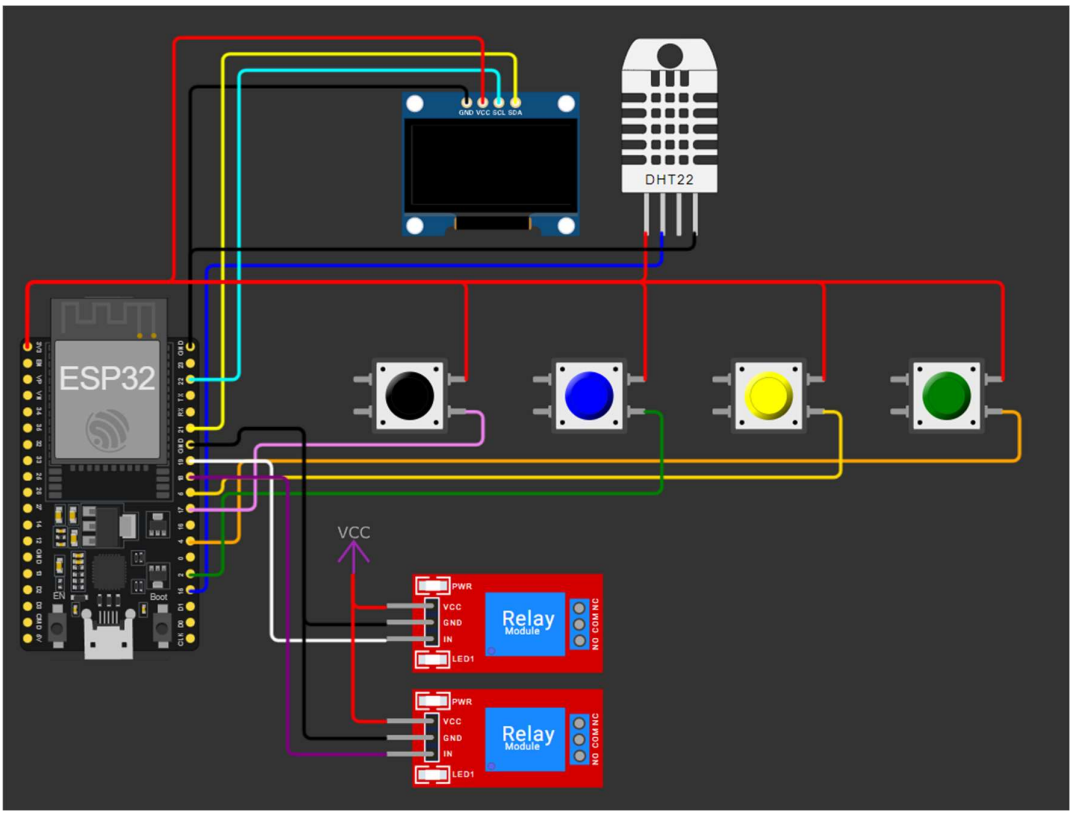
III. **Gráfica de temperatura**:

- 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.

IV. **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

Este proyecto proporciona una solución eficiente y accesible para el secado de filamento, asegurando un control estable de temperatura y humedad. La implementación de un ESP32 permite una gestión inteligente y la posibilidad de monitoreo remoto.

Como futuras mejoras:

- Seguridad, como protecciones térmicas frente a sobrecalentamiento de la resistencia
- Alimentación directa de impresoras, esto permite preservar el filamento seco mientras se imprime