

PROYECTO HELIOS - HASP 2024

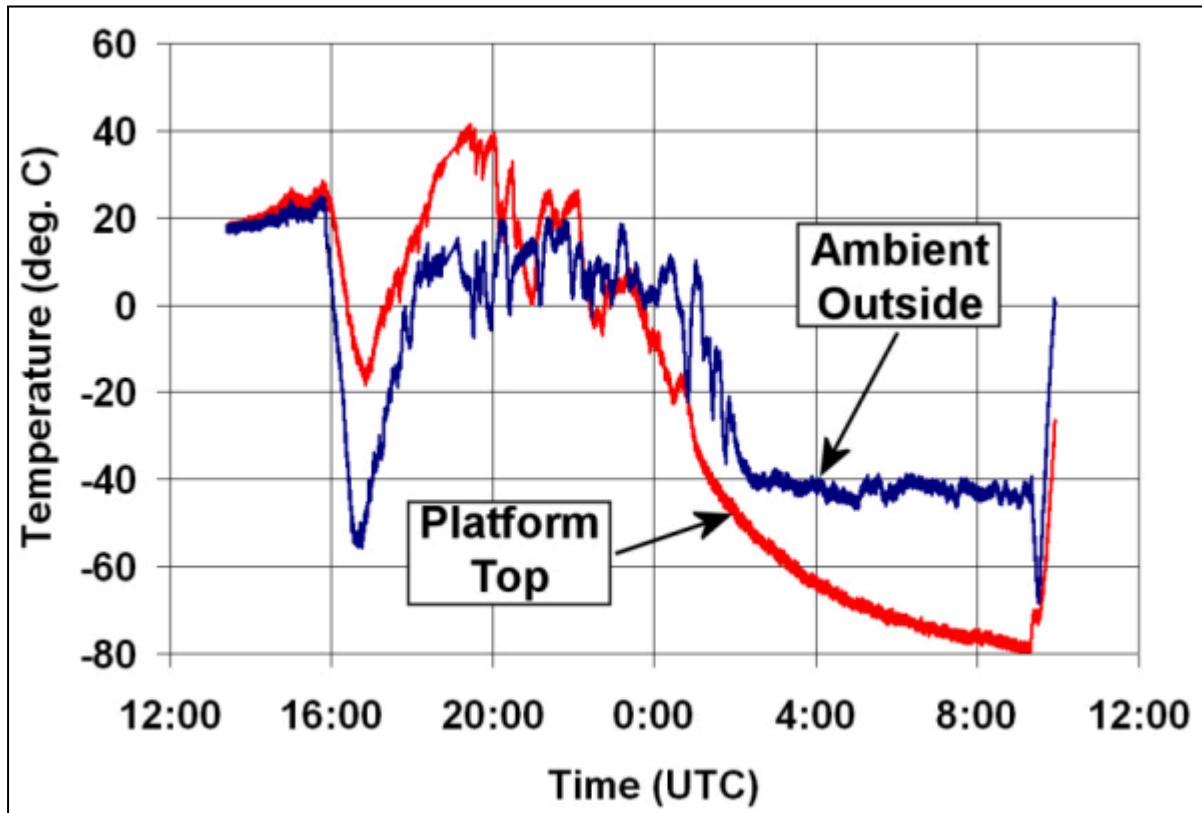
CONTROL DE TEMPERATURA

David Fernando Evangelista Cuti

I. OBJETIVO

El objetivo del control térmico es mantener la temperatura adecuada y estable del payload, garantizando que sus componentes internos operen dentro de un rango óptimo para asegurar su correcto funcionamiento.

Figura 1. Temperaturas típicas durante el vuelo



Según se muestra en la Figura 1, se observa el rango de temperaturas al que se somete el payload al alcanzar grandes altitudes durante el vuelo. Como se mencionó, el objetivo es mantener la temperatura dentro de un rango estable de entre 20 y 25 °C.

La primera propuesta es desarrollar un control térmico utilizando un controlador difuso, ya que es capaz de modelar comportamientos complejos y no lineales de manera eficiente, sin necesidad de una modelización matemática exacta del sistema.

II. CONTROL DIFUSO DE TEMPERATURA INICIAL

Para implementar un control difuso, es fundamental comprender los conceptos principales, lo que permitirá gestionar adecuadamente las variables de temperatura necesarias para el control.

Funciones de membresía

Son componentes clave en la lógica difusa. Representan gráficamente el grado de pertenencia de un valor a un conjunto difuso, asignando un valor entre 0 y 1 a cada posible entrada.

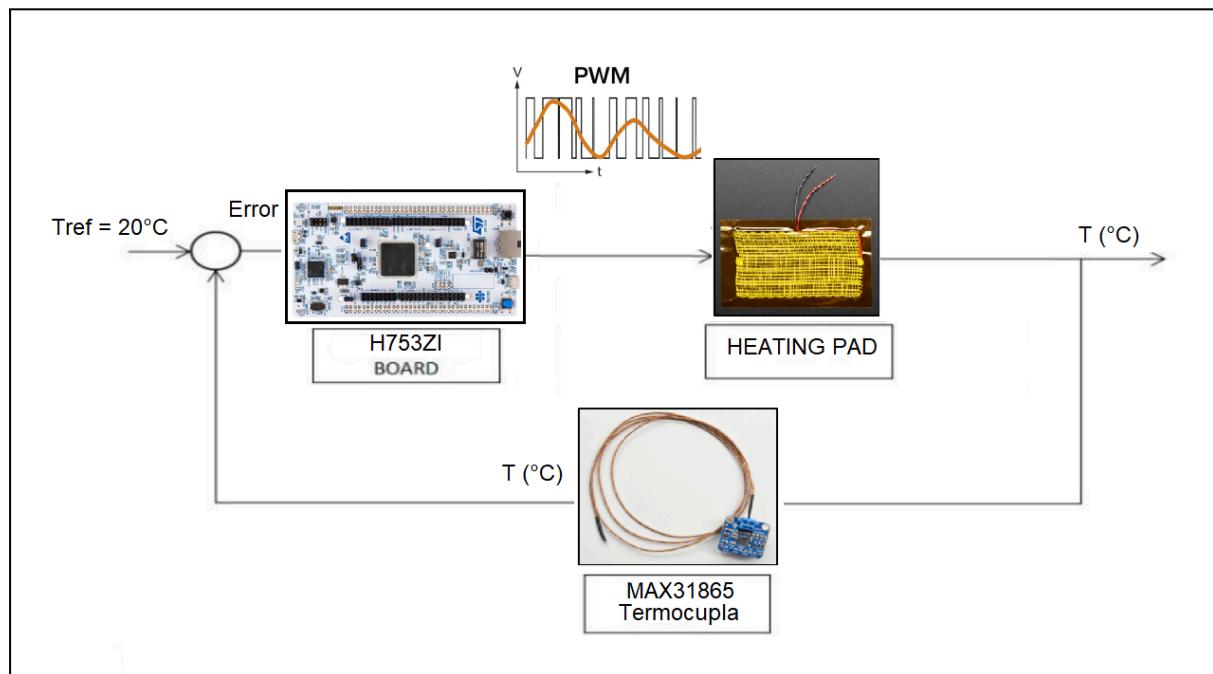
Reglas de control (asignación de funciones de membresía)

Son directrices que determinan cómo debe actuar un sistema de control en función de las condiciones actuales.

Método de inferencia Mamdani

El método de inferencia de Mamdani es uno de los enfoques más comunes en los sistemas de lógica difusa para tomar decisiones. Se basa en reglas "si-entonces" y utiliza funciones de membresía para procesar las entradas difusas.

Figura 2. Diagrama básico



En la Figura 2 se presenta el sistema de control retroalimentado, en el cual se utiliza un controlador implementado en una tarjeta STM32 modelo H753ZI. Los *heating pads* actúan como actuadores, mientras que se emplean termocuplas como sensores de temperatura, acompañadas de su módulo de lectura. Los datos de estos sensores se envían nuevamente al microcontrolador de manera retroalimentada, con el fin de reducir el error y alcanzar una temperatura estable.

Es importante destacar que los *heating pads* serán controlados mediante una señal PWM, lo que permitirá modificar su temperatura de manera eficiente.

Tabla 1. Características del control de temperatura

T, Rango de temperatura de análisis	[-80; 40] °C
Tref, Temperatura deseada	20°C
Error	Tref - T, [100; -20]
Función de entrada	[100; -20]
Función de salida (PWM)	[0; 100]

En la Tabla 1 se pueden visualizar los parámetros calculados para la implementación del controlador. Para el desarrollo del control, se utilizó el programa MATLAB como herramienta de prueba, con el objetivo de evaluar la eficiencia y determinar la viabilidad de implementar el controlador.

i. Implementación en MATLAB Simulink (pruebas)

Figura 3. Desarrollo del controlador difuso

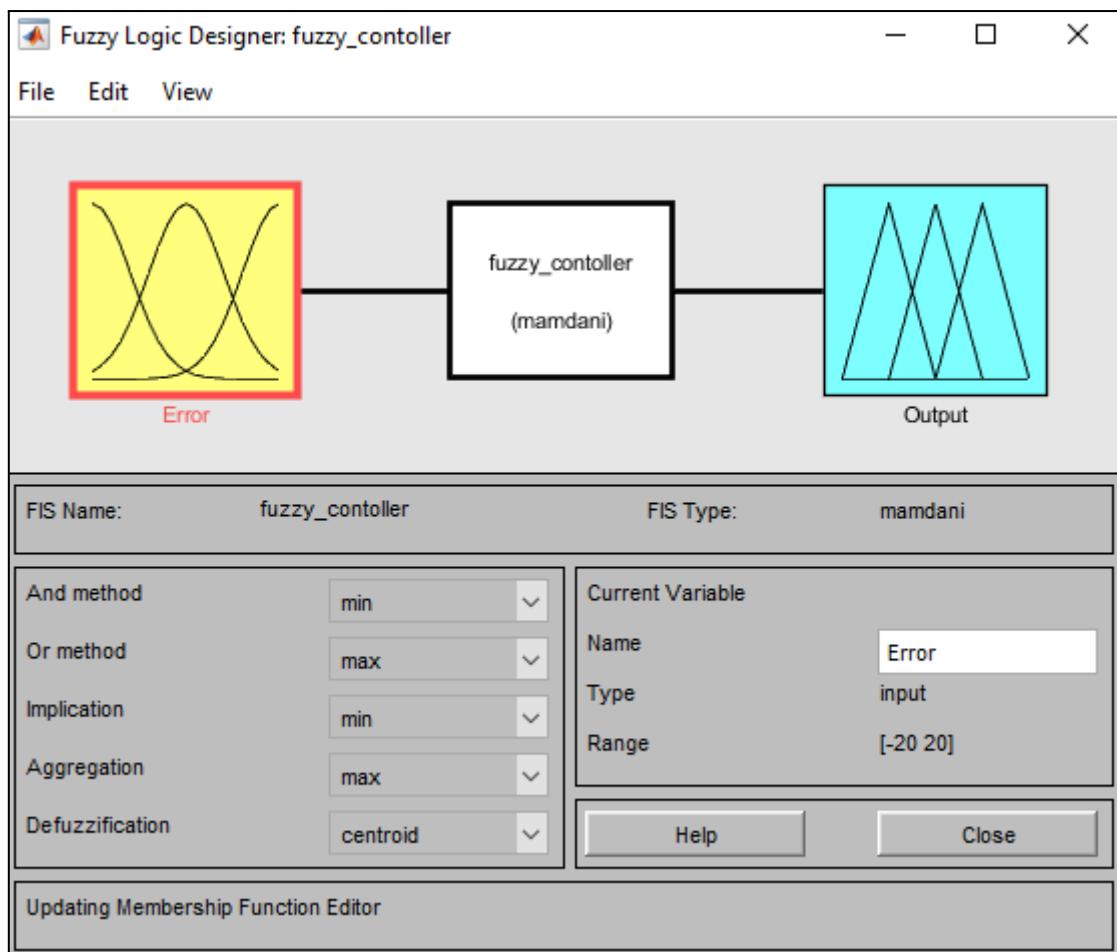


Figura 4. Implementación de las funciones de membresía para el error

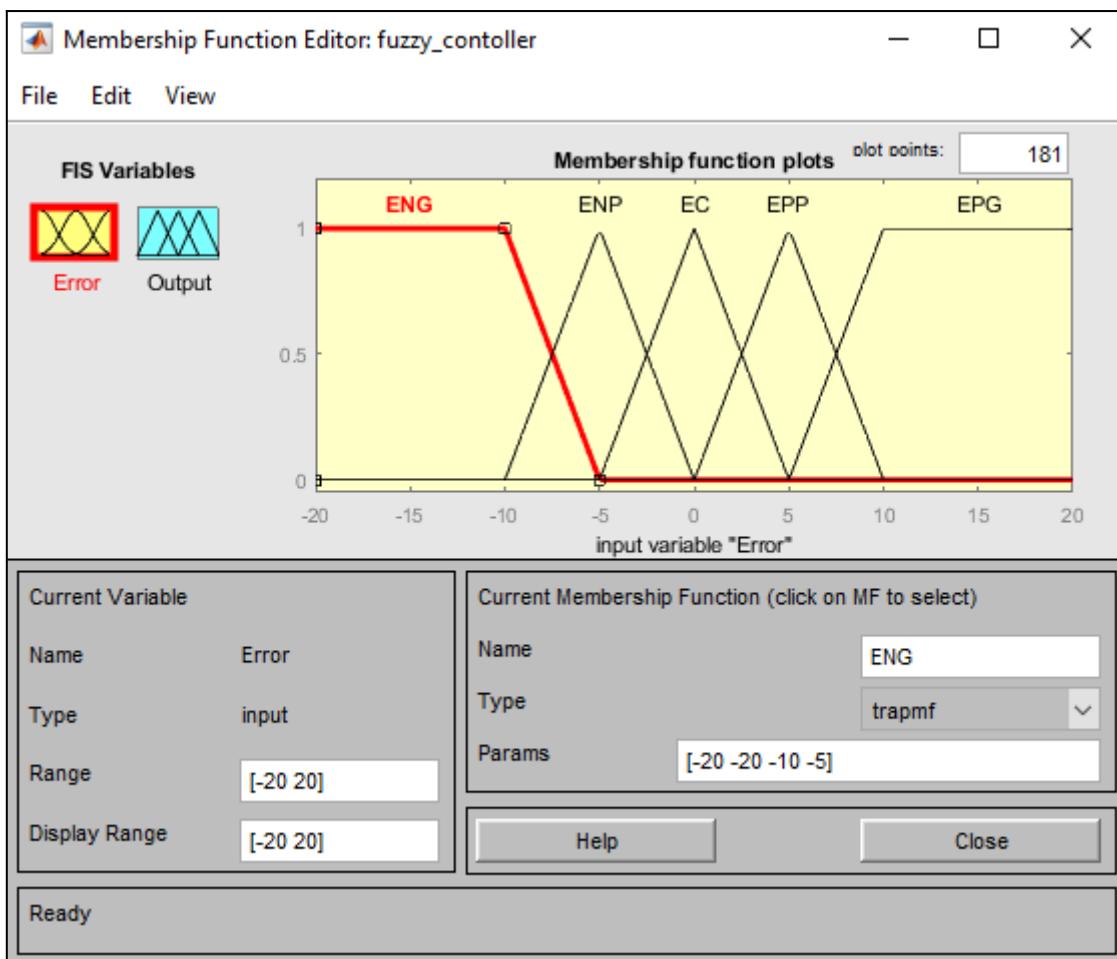


Figura 5. Implementación de las funciones de membresía para la salida

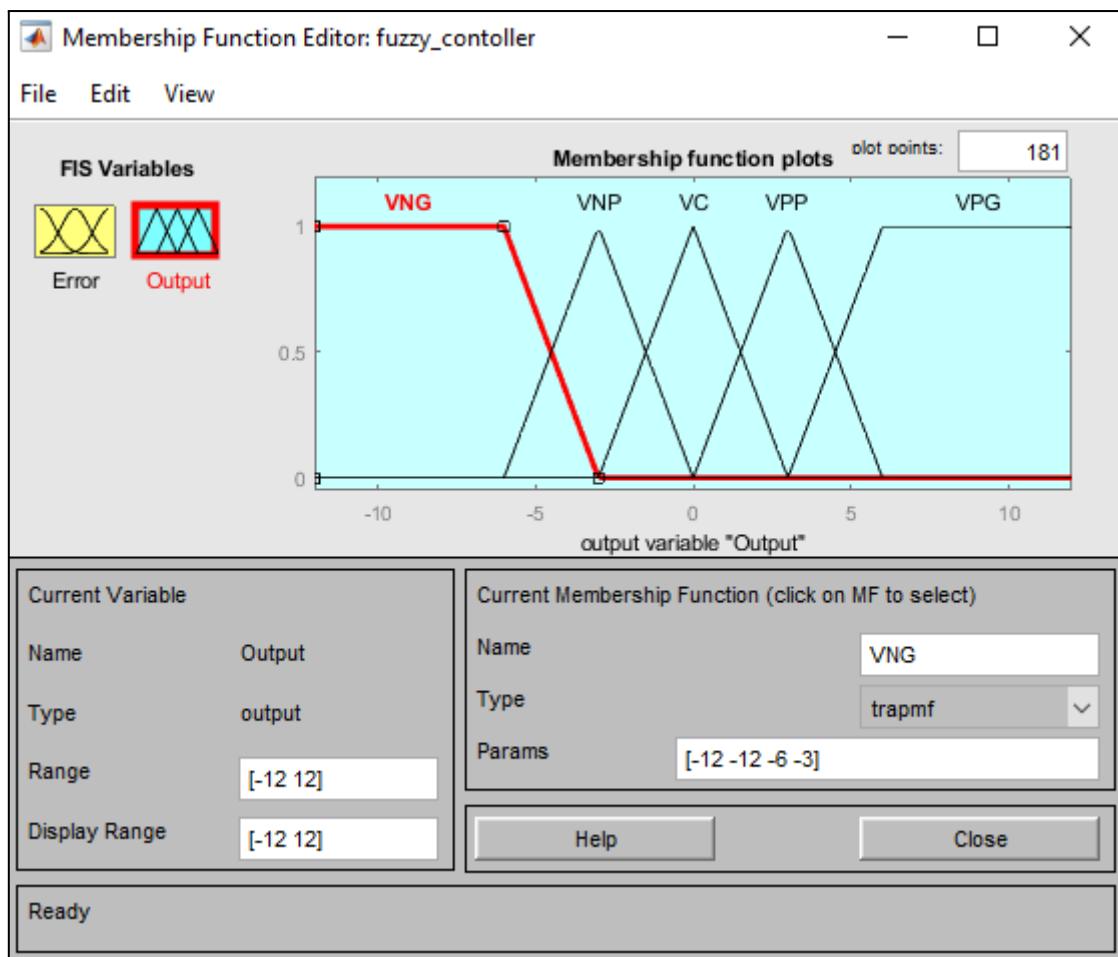
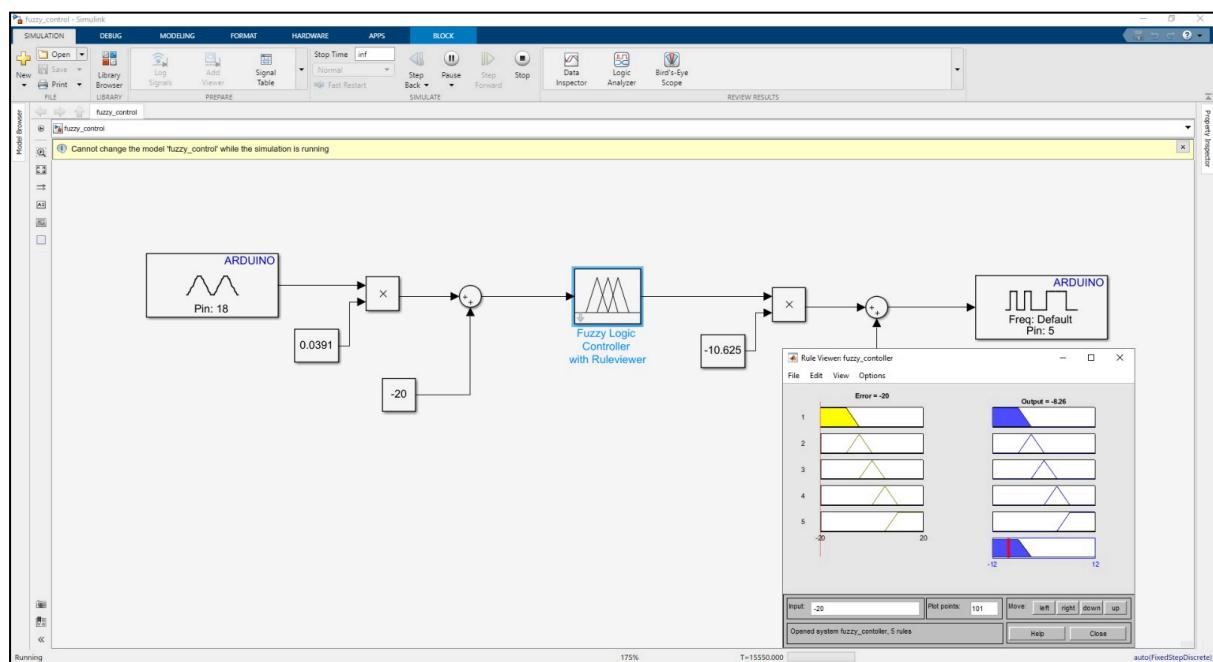
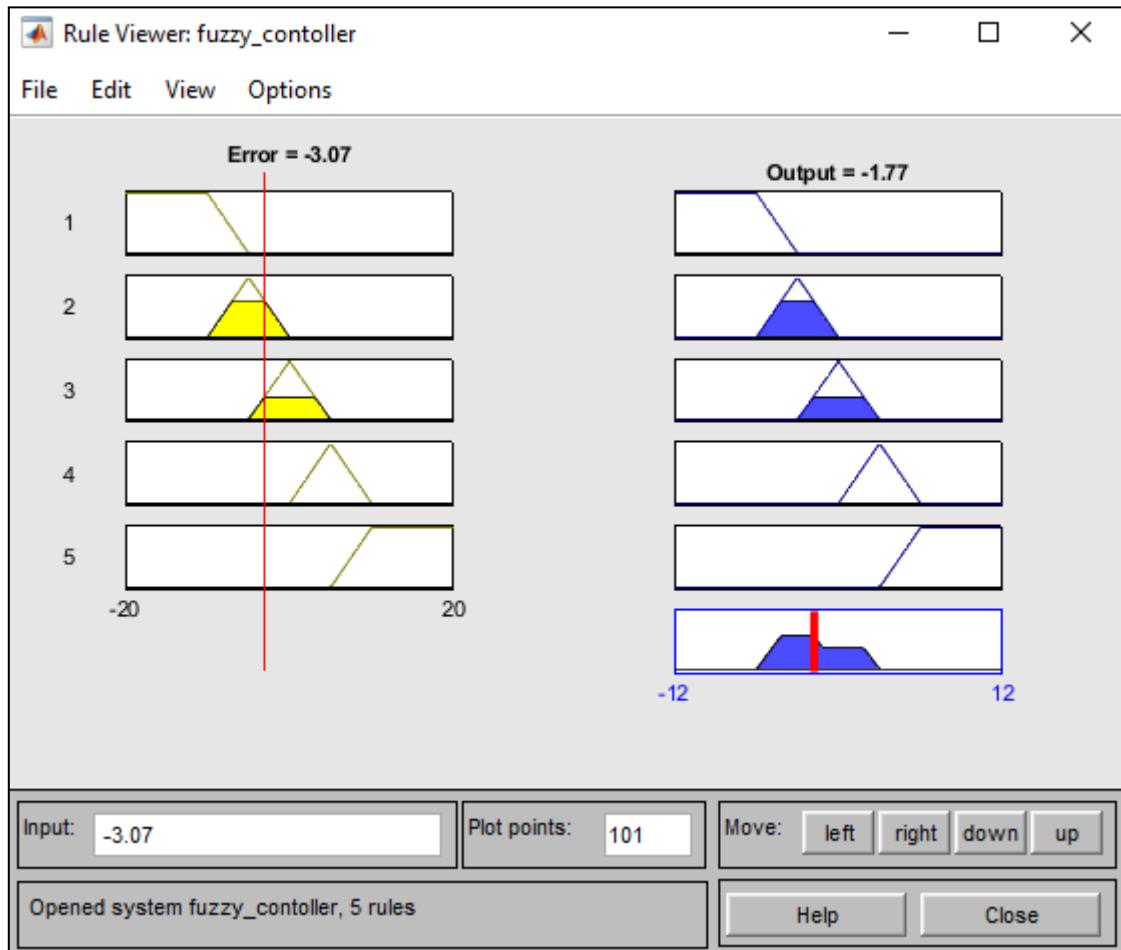


Figura 6. Sistema de control implementado en MATLAB



En la Figura 6 se presenta el sistema completo que utiliza el control difuso. El microcontrolador empleado para la prueba es un Arduino. La salida es una señal PWM que varía entre 0 y 100% del valor máximo que puede entregar un pin del Arduino, en función de la variación de los parámetros suministrados al controlador.

Figura 7. Variación de los parámetros de control



ii. Desarrollo del programa de control difuso

Para el desarrollo del controlador, se utilizó el IDE de Arduino. Se emplearon diversas bibliotecas para implementar el control, como se muestra en la Figura 8.

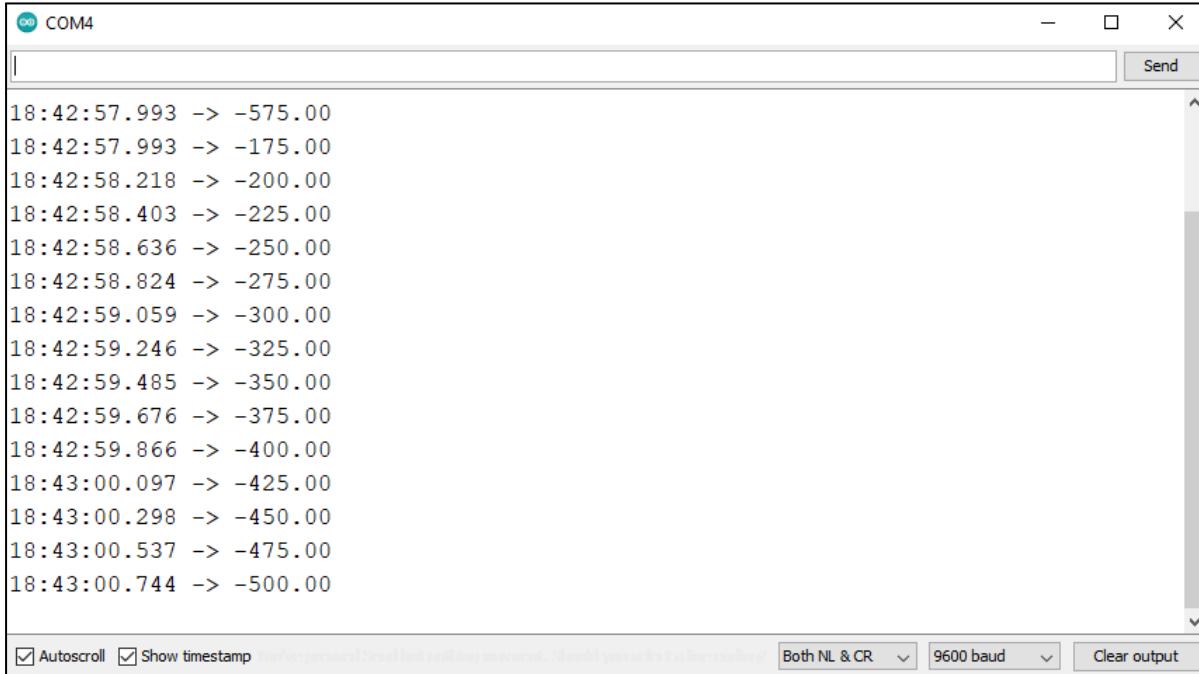
Figura 8. Librería fuzzy

	examples	09/08/2019 08:57 p. m.	Carpeta de archivos
	fuzzy	25/11/2019 04:07 p. m.	C++ Source File
	fuzzy	25/11/2019 04:08 p. m.	Archivo H

Figura 9. Programa desarrollado empleando la librería *fuzzy*

```
29 void loop() {
30   float B[tam]; //Creamos el vector que guardará los numeros del conjunto de salida
31   controlfuzzy.inicio(B,tam); //Inicializamos el objeto fuzzy
32   error=setpoint-rpm;//Calculamos la variable error
33   /*
34    * Se empieza con la inferencia, y para ello las reglas. Para llamar a una regla con una entrada se utiliza la función "regla_simple"
35    * la cual recibe como parametros (Conjunto entrada,Universo de entrada,variable a evaluar,conjunto salida,universo salida,vector salida,tamaño
36    * vector salida)
37   */
38   controlfuzzy.regla_simple(ENP,UIN,error,DT,USAL,B,tam);//Llamamos a la función para aplicar la regla correspondiente
39   controlfuzzy.regla_simple(EC,UIN,error,ZE,USAL,B,tam);
40   controlfuzzy.regla_simple(EPP,UIN,error,AT,USAL,B,tam);
41   float res = controlfuzzy.defusify(B,USAL,tam); //Se llama a la función para defusificar el conjunto de salida y obtener el resultado del sistema fuzzy
42   rpm = rpm+res; // Se asigna ese resultado a la variable control.
43   Serial.println(rpm); //Se imprime el valor de rpm que debería converger al valor de setpoint
44   delay(200);
45   /*
46    * En este ejemplo el valor de rpm debería seguir el valor definido en setpoint
47   */
48 }
```

Figura 10. Resultados de funcionamiento del controlador



```
18:42:57.993 -> -575.00
18:42:57.993 -> -175.00
18:42:58.218 -> -200.00
18:42:58.403 -> -225.00
18:42:58.636 -> -250.00
18:42:58.824 -> -275.00
18:42:59.059 -> -300.00
18:42:59.246 -> -325.00
18:42:59.485 -> -350.00
18:42:59.676 -> -375.00
18:42:59.866 -> -400.00
18:43:00.097 -> -425.00
18:43:00.298 -> -450.00
18:43:00.537 -> -475.00
18:43:00.744 -> -500.00
```

ANEXOS

Anexo 1. Librería *fuzzy.cpp*

```
#include <Arduino.h>
#include "fuzzy.h"

void fuzzy::inicio(float B[],int tama) {
  for(int i=0; i<tama; i++) {
    B[i] = 0;
```

```

    }

}

void fuzzy::borrar(float B[],int tama) {
    for(int i=0; i<tama; i++) {
        B[i] = 0;
    }
}

int fuzzy::calc_size(float vec[], float paso) {
    int res = 0;
    res = ((absoluto(vec[0]))+absoluto((vec[1])))/paso;
    return res;
}

void fuzzy::regla_simple(float in[],float Uin[],float err,float V[], float
UV[], float B[], int tam) {
    int t=0,date=0;
    float r=0.0;
    while(date==0) {
        if(in[t]=='T' || in[t]=='t' || in[t]=='R' || in[t]=='r') {
            date=1;
            t=t-1;
        }
        t=t+1;
    }
    if(t ==3) {
        r = triangular(Uin,in,err);
    }
    else if(t ==4) {
        r = trapezoidal(Uin,in,err);
    }
    inferencia_mamdani2(B,r,V,UV,tam); // Se Hace la primera regla
}

void fuzzy::regla_compuesta2(float in1[],float in2[],float U1[],float
U2[],float c1,float c2, float V[], float UV[], float B[], int tam) {
    int t1=0,date = 0,t2 =0,date1=0;
    float r1 = 0, r2 = 0,res =0;
    while(date ==0) {
        while(date1==0) {
            if(in2[t2]=='T' || in2[t2]=='t' || in2[t2]=='R' || in2[t2]=='r') {
                date1=1;
                t2=t2-1;
            }
        }
    }
}

```

```

        t2=t2+1;
    }
    if(in1[t1]=='T' || in1[t1]=='t' || in1[t1]=='R' || in1[t1]=='r') {
        date=1;
        t1=t1-1;
    }
    t1=t1+1;//aqui defino el tamaño de in1 e in2
}
if(t1==3) {
    r1 = triangular(U1,in1,c1);
}
else if(t1==4) {
    r1 = trapezoidal(U1,in1,c1);
}
if(t2==3) {
    r2 = triangular(U2,in2,c2);
}
else if(t2 ==4) {
    r2 = trapezoidal(U2,in2,c2);
}
res = min(r1,r2);
inferencia_mamdani2(B,res,V,UV,tam);
}

void fuzzy::regla_compuesta3(float in1[],float in2[],float in3[],float
U1[],float U2[],float U3[],float c1,float c2,float c3, float V[], float UV[],
float B[], int tam){
int t1=0,date = 0,t2 = 0,t3 = 0,date1=0,date2 = 0;
float r1 = 0, r2 = 0,r3 =0,res =0;
while(date ==0) {
    while(date1==0) {
        while(date2==0) {
            if(in3[t3]=='T' || in3[t3]=='t' || in3[t2]=='R' || in3[t3]=='r') {
                date2=1;
                t3=t3-1;
            }
            t3=t3+1; //Aqui defino el tamaño de in3
        }
        if(in2[t2]=='T' || in2[t2]=='t' || in2[t2]=='R' || in2[t2]=='r') {
            date1=1;
            t2=t2-1;
        }
        t2=t2+1;//aqui defino el tamaño de in2
    }
    if(in1[t1]=='T' || in1[t1]=='t' || in1[t1]=='R' || in1[t1]=='r') {

```

```

        date=1;
        t1=t1-1;
    }
    t1=t1+1;//aqui defino el tamaño de in1
}

if(t1==3){
    r1 = triangular(U1,in1,c1);
}
else if(t1==4){
    r1 = trapezoidal(U1,in1,c1);
}
if(t2==3){
    r2 = triangular(U2,in2,c2);
}
else if(t2 ==4){
    r2 = trapezoidal(U2,in2,c2);
}
if(t3==3){
    r3 = triangular(U3,in3,c3);
}
else if(t3 ==4){
    r3 = trapezoidal(U3,in3,c3);
}
res = min(r1,r2);
res = min(r3,res);
inferencia_mamdani2(B,res,V,UV,tam);
}

float fuzzy::absoluto(float dat){// Función utilizada para calcular el
absoluto de un numero
    float res;
    if(dat <0){
        res = dat*(-1);
    }
    else{
        res = dat;
    }
    return res;
}

float fuzzy::triangular(float x[],float params[],float date){
    float y;
    if(date < params[0]){ // Se rellena con ceros a la izquierda
        y = 0;
    }
    else if(date > params[1]){
        y = 1;
    }
    else{
        y = ((date - params[0])/(params[1] - params[0])) * (1 - ((date - params[0])/(params[1] - params[0])));
    }
    return y;
}

```

```

}

if(date > params[2]) { // Se rellena con ceros a la derecha
    y = 0;
}

if (params[0]<=date && date<= params[1]) { // Primera Pendiente
    float p = 1/(params[1]-params[0]);
    y = (p*(date-params[1])+1);
}

if (params[1]<=date && date<= params[2]) { // Segunda Pendiente
    float p = -1/(params[2]-params[1]);
    y = (p*(date-params[1])+1);
}

if(params[0] <x[0] || params[2] >x[1]){
    y = 0;
}

if((date == params[0]&& date ==params[1])|| (date == params[1]&& date
==params[2])){
    y = 1;
}

return y;
}

float fuzzy::trapezoidal(float x[],float params[],float date) {
    float y;

    if(date < params[0]) { // Se rellena con ceros a la izquierda
        y = 0;
    }

    if(date > params[2]) { // Se rellena con ceros a la derecha
        y = 0;
    }

    if (params[0]<=date && date<= params[1]) { // Primera Pendiente
        float p = 1/(params[1]-params[0]);
        y = (p*(date-params[1])+1);
    }

    if (params[1]<date && date<= params[2]) { // Región Constante
        y = 1;
    }

    if (params[2]<=date && date<= params[3]) { // Segunda Pendiente
        float p = -1/(params[3]-params[2]);
        y = (p*(date-params[2])+1);
    }

    if(params[0] <x[0] || params[2] >x[1]){
        y = 0;
    }
}

```

```

if((date == params[2]&& date ==params[3]) || (date == params[0]&& date
==params[1]) || (date == params[1]&& date ==params[2])) {
    y = 1;
}
return y;
}

void fuzzy::inferencia_mamdani2(float B[],float c,float stru[],float
U_VOL[],int tama){ //Hacemos inferencia mamdani
//float j=-12; //Inicio del universo de discurso de voltaje
int tipo,t=0,date = 0;
while(date ==0) {
    if(stru[t]=='T' || stru[t]=='t' || stru[t]=='R' || stru[t]=='r') {
        date=1;
        t=t-1;
    }
    t=t+1;
}
float paso,j=U_VOL[0];
paso = (absoluto(U_VOL[0])+absoluto(U_VOL[1]))/float(tama);
//Serial.println(paso,6);
if(t ==3)
{
    tipo = 2; //Función Triangular
}
else if(t ==4)
{
    tipo =1; //Función tipo trapezoidal
}
float v[tama]; // Tamaño del vector del universo de discurso de voltaje
for(int i=0; i <tama;i++){ //Aqui genero vector v
    v[i] = j;
    j = j+paso;
}
if(tipo ==1){
    for(int i =0;i<tama;i++){
        if(v[i]<stru[0] || v[i]==stru[0]){
            float y=0;
            if(y>B[i]){
                B[i]=y;
            }
        }
        if(stru[0]<= v[i] && v[i]<=stru[1]){ //Primera Pendiente
            float p= 1/(stru[1]-stru[0]);
            float y = (p*(v[i]-stru[0]))+1;
            B[i]=y;
        }
    }
}

```

```

    if(y>=c) {
        y=c;
    }
    if(y>B[i]) {
        B[i]=y;
    }
}
if(stru[1]<=v[i] && v[i]<=stru[2]) { //zona constante
    float y=c;
    if(y>B[i]) {
        B[i]=y;
    }
}
if(stru[2]<=v[i] && v[i]<=stru[3]) { //Segunda pendiente
    float p= -1/(stru[3]-stru[2]);
    float y= (p*(v[i]-stru[2]))+1;
    if(y>=c) { //Cortamos a la altura de c1
        y=c;
    }
    if(y>B[i]) {
        B[i]=y;
    }
}
if(v[i]==stru[2] && v[i]==stru[3]) { //ultimos puntos iguales
    float y=c;
    if(y>B[i]) {
        B[i]=y;
    }
}
if(v[i]>stru[3]) {
    float y=0;
    if(y>B[i]) {
        B[i]=0;
    }
}
}

if(tipo ==2) {
for(int i=0; i<tama; i++) {
    if(stru[0]<= v[i] && v[i]<=stru[1]) { //Primera Pendiente
        float p= 1/(stru[1]-stru[0]);
        float y= (p*(v[i]-stru[1]))+1;
        if(y >=c){ //Esto es para hacer el recorte a la altura del valor de
membresia
}
}
}
}

```

```

        y=c;
    }
    if(y >B[i]) { //Esto es para sacar el máximo, unir los conjuntos
        B[i]=y;
        //Serial.println(i);
    }
}
if(stru[1]<=v[i] && v[i]<=stru[2]) { //Segunda pendiente
    float p= -1/(stru[2]-stru[1]);
    float y = (p*(v[i]-stru[1]))+1;
    if(y>=c) { //Hacemos el corte a la altura del valor de membresia
        y=c;
    }
    if(y>B[i]) { //Comparamos los valores de membresia
        B[i]=y;
    }
}
if(v[i] ==stru[1] && v[i]==stru[2]) { //Cuando los valores 2 y 3 son
iguales
    float y=0; ////////ESTO ESTA SUJETO A CAMBIOS
}
}
}

float fuzzy::defusi(float conju[],float U_VOL[],int tama){
float j=U_VOL[0],paso;
paso = (absoluto(U_VOL[0])+absoluto(U_VOL[1]))/tama;
float x[tama];
float y0,area=0,res1=0;
for(int i=0; i <tama;i++){
    x[i] = j;
    j = j+paso;
    area= area+conju[i];
    res1= res1+(conju[i]*x[i]);
    conju[i]=0;
}
if(area ==0) {
    y0 = 0;
}
else{
    y0 = res1/area;
}
return y0;
}

```

Anexo 2. Librería *fuzzy.h*

```
#ifndef fuzzy_h
#define fuzzy_h
#include <arduino.h>
class fuzzy
{
public:
    int calc_size(float vec[], float paso);
    float triangular(float x[], float params[], float date);
    float trapezoidal(float x[], float params[], float date);
    void inferencia_mamdani2(float B[], float c, float stru[], float
U_VOL[], int tam);
    float defusi(float conju[], float U_VOL[], int tam);
    float absoluto(float dat);
    void regla_compuesta2(float in1[], float in2[], float U1[], float
U2[], float c1, float c2, float V[], float UV[], float B[], int tam);
    void regla_compuesta3(float in1[], float in2[], float in3[], float
U1[], float U2[], float U3[], float c1, float c2, float c3, float V[], float
UV[], float B[], int tam);
    void regla_simple(float in[], float Uin[], float err, float V[], float
UV[], float B[], int tam);
    void inicio(float B[], int tama);
    void borrar(float B[], int tama);
};
#endif
```

Anexo 3. Programa Principal

```
#include <fuzzy.h> // Incluimos la librería
fuzzy controlfuzzy; // Creamos un objeto de tipo fuzzy.
//////////////////////////////Definimos
//////////////////////////////conjunto para variable de entrada
float USAL[] = {-50,50}; // Definimos el universo de discurso (rango) de
la variable salida
float UIN[] = {-1500,1500}; // Definimos universo de discurso de la
variable de entrada
float ENP[] = {-1500,-666,0,'T'}; //Definimos los conjuntos difusos,
colocando 'T' si es triangular, o 'R' si es trapezoidal
float EC[] = {-5,0,5,'T'};// Conjuntos difusos de entrada
```

```

float EPP[] = {6,666,1500,'T'};
/////////////////////////////// Definimos
/////////////////////////////// conjuntos variable de salida
float DT[] = {-40,-30,-20,-10,'R'}; //Conjunto difuso trapezoidal
float ZE[] = {-10,0,10,'T'}; //Conjunto difuso triangular
float AT[] = {10,20,30,40,'R'};// Conjunto difuso trapezoidal
float paso = 2; //Se define la cantidad mínima de variación para los
numeros del conjunto de salida
float setpoint =1000,rpm=0,error;
const int tam = controlfuzzy.calc_size(USAL,paso); // Calculamos el
tamaño que tendrá el vector del conjunto de salida con universo salida
y "paso"
void setup() { //Configuración
Serial.begin(9600);
}
///////////////////////////////
/////////////////////////////
void loop() {
    float B[tam]; //Creamos el vector que guardará los numeros del
conjunto de salida
    controlfuzzy.inicio(B,tam); //Inicializamos el objeto fuzzy
    error=setpoint-rpm;//Calculamos la variable error
/*
 * Se empieza con la inferencia, y para ello las reglas. Para llamar a
una regla con una entrada se utiliza la función "regla_simple"
 * la cual recibe como parametros (Conjunto entrada,Universo de
entrada,variable a evaluar,conjunto salida,universo salida,vector
salida,tamaño
 * vector salida)
 */
controlfuzzy.regla_simple(ENP,UIN,error,DT,USAL,B,tam);//Llamamos a la
función para aplicar la regla correspondiente
controlfuzzy.regla_simple(EC,UIN,error,ZE,USAL,B,tam);
controlfuzzy.regla_simple(EPP,UIN,error,AT,USAL,B,tam);
float res = controlfuzzy.defusi(B,USAL,tam); //Se llama a la función
para defusificar el conjunto de salida y obtener el resultado del
sistema fuzzy
rpm = rpm+res; // Se asigna ese resultado a la variable control.
Serial.println(rpm); //Se imprime el valor de rpm que debería
converger al valor de setpoint
delay(50);
/*

```

```

    * En este ejemplo el valor de rpm debería seguir el valor definido en
    setpoint
    */
}

```

III. CONTROL DIFUSO DE TEMPERATURA TARJETA STM32

Como segunda fase del desarrollo del controlador, se optó por utilizar la tarjeta NUCLEO-H753ZI. No obstante, las pruebas iniciales se llevaron a cabo con la tarjeta NUCLEO-L432KC para evaluar este microcontrolador. El IDE utilizado es **STM32CubeIDE 1.14.1**.

Figura 11. Pinout del NUCLEO-L432KC

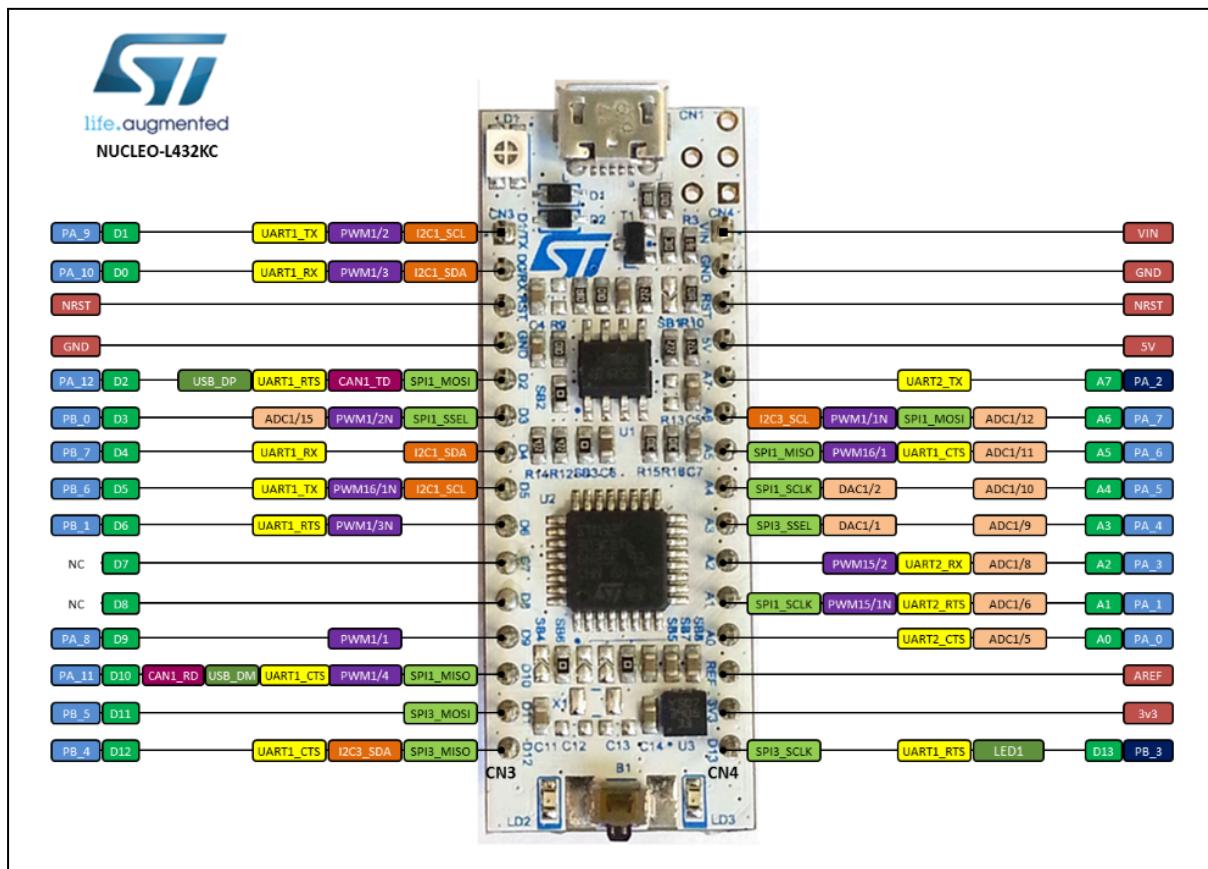
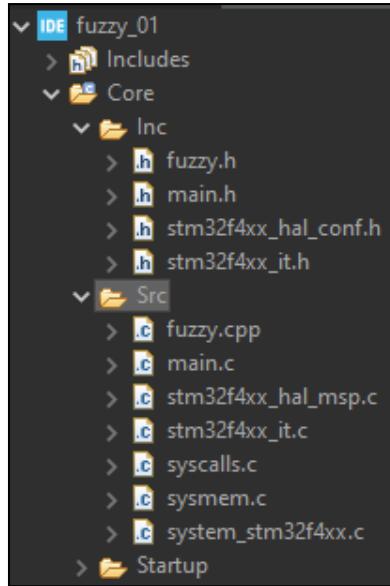


Figura 12. Proyecto desarrollado en el IDE



iii. Pruebas de librerías

Para el NUCLEO-L432KC se realizo las siguientes revisiones de librerías

- 1) <https://github.com/alvesoaj/eFLL>
- 2) https://github.com/Jaroan/Fuzzy_PID_KRSSG

Lcd

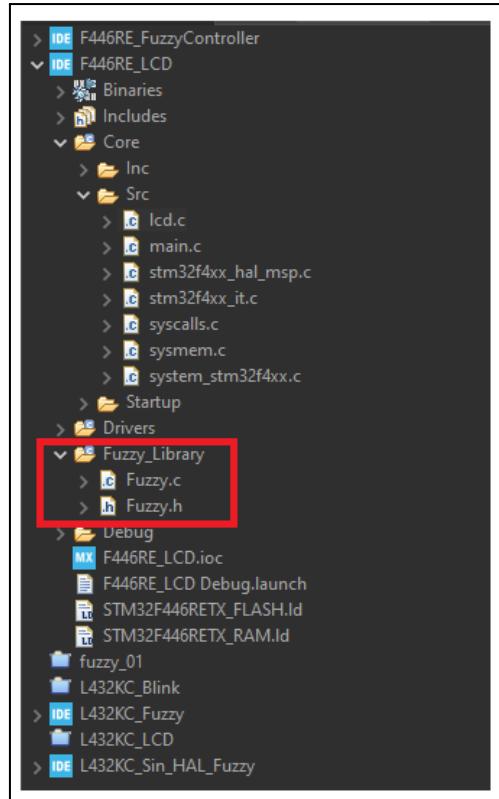
El LCD funciona con alimentación USB, y sus conexiones de alimentación están conectadas al alimentador de la protoboard, con el ánodo a 3.3V. Los pines de datos se conectan de acuerdo con el enlace: <https://www.micropeta.com/video60>.

La ubicación del código principal para el manejo del LCD se encuentra en el programa F446_LCD. Es recomendable dejar el programa en ejecución durante un tiempo y luego volver al modo de depuración.

Controlador

El controlador difuso utiliza como entradas el error y el cambio en el error, así como las salidas Kp y Kd. Estos nuevos valores se introducen en una función PD que calcula el PWM requerido.

Figura 13. Librerías utilizadas



En la Figura 13 se pueden observar los programas de las bibliotecas: *Fuzzy.c* y *Fuzzy.h*. Cabe recordar que estas se desarrollaron a partir de la biblioteca para Arduino, siendo trasladadas del lenguaje C++ al lenguaje C.

ANEXOS

L432KC_LCD *main.c*

```
/* USER CODE BEGIN Header */
/** 
 * ***** 
 * @file          : main.c
 * @brief         : Main program body
 * 
 * ***** 
 * @attention
 *
 * Copyright (c) 2024 STMicroelectronics.
 */
```

```
* All rights reserved.  
*  
* This software is licensed under terms that can be found in the  
LICENSE file  
* in the root directory of this software component.  
* If no LICENSE file comes with this software, it is provided AS-IS.  
*  
*****  
*****  
*/  
/* USER CODE END Header */  
/* Includes  
-----*/  
#include "main.h"  
  
/* Private includes  
-----*/  
/* USER CODE BEGIN Includes */  
#include "lcd.h"  
#include <stdio.h>  
/* USER CODE END Includes */  
  
/* Private typedef  
-----*/  
/* USER CODE BEGIN PTD */  
  
/* USER CODE END PTD */  
  
/* Private define  
-----*/  
/* USER CODE BEGIN PD */  
  
/* USER CODE END PD */  
  
/* Private macro  
-----*/  
/* USER CODE BEGIN PM */  
  
/* USER CODE END PM */  
  
/* Private variables  
-----*/
```

```
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes
-----
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code
-----
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU
Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */
    Lcd_Init();
    /* USER CODE END Init */

    /* Configure the system clock */

```

```

SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
/* USER CODE BEGIN 2 */
Lcd_Clear();
/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
    Lcd_Set_Cursor(1, 1);
    Lcd_Send_String("Hello");
    HAL_Delay(500);
    Lcd_Clear();

}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

/** Configure the main internal regulator output voltage
*/
if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
{

```

```

        Error_Handler();
    }

/** Configure LSE Drive Capability
*/
HAL_PWR_EnableBkUpAccess();
__HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);

/** Initializes the RCC Oscillators according to the specified
parameters
 * in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_LSE|RCC_OSCILLATORTYPE_MSI;
RCC_OscInitStruct.LSEState = RCC_LSE_ON;
RCC_OscInitStruct.MSISState = RCC_MSI_ON;
RCC_OscInitStruct.MSICalibrationValue = 0;
RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
RCC_OscInitStruct.PLL.PLLM = 1;
RCC_OscInitStruct.PLL.PLLN = 16;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                           |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) !=
HAL_OK)
{
    Error_Handler();
}

```

```

}

/** Enable MSI Auto calibration
 */
HAL_RCCEx_EnableMSIPLLMode();
}

/** @brief USART2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART2_UART_Init(void)
{

/* USER CODE BEGIN USART2_Init_0 */

/* USER CODE END USART2_Init_0 */

/* USER CODE BEGIN USART2_Init_1 */

/* USER CODE END USART2_Init_1 */
huart2.Instance = USART2;
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
if (HAL_UART_Init(&huart2) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN USART2_Init_2 */

/* USER CODE END USART2_Init_2 */
}

/** @*/

```

```

* @brief GPIO Initialization Function
* @param None
* @retval None
*/
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOB, D6_Pin|D7_Pin|LD3_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOA, RS_Pin|E_Pin|D4_Pin|D5_Pin, GPIO_PIN_RESET);

/*Configure GPIO pins : D6_Pin D7_Pin LD3_Pin */
GPIO_InitStruct.Pin = D6_Pin|D7_Pin|LD3_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pins : RS_Pin E_Pin D4_Pin D5_Pin */
GPIO_InitStruct.Pin = RS_Pin|E_Pin|D4_Pin|D5_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */
/* USER CODE END 4 */

```

```

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return
     state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifndef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line
 * number
 *           where the assert_param error has occurred.
 * @param   file: pointer to the source file name
 * @param   line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and
     line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
     line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```

L432KC_Fuzzy *main.c*

```

/* USER CODE BEGIN Header */
/**
```

```
*****
* @file          : main.c
* @brief         : Main program body

*****
* @attention
*
* Copyright (c) 2024 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the
LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.
*

*****
*/
/* USER CODE END Header */
/* Includes
-----*/
#include "main.h"

/* Private includes
-----*/
/* USER CODE BEGIN Includes */
#include <stdlib.h>
#include <stdio.h>
/* USER CODE END Includes */

/* Private typedef
-----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define
-----*/
/* USER CODE BEGIN PD */
```

```

/* USER CODE END PD */

/* Private macro
-----
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables
-----
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

///////////
///////
volatile int Fuzzy_Matrix[3][3];
volatile int
NE,ND,E,PDE,ZE,ZDE,err_sum,error,d_err,Kp,Ki,Kd,prev_err;
//volatile float
volatile int PWM_Fuzzy,ticks;
volatile int target=50;
volatile int Kp_small,Kp_medium,Kp_large;
volatile int Ki_small,Ki_medium,Ki_large;
volatile int Kd_small,Kd_medium,Kd_large;
volatile int out;
const int Kp_s=1,Kp_m=2,Kp_l=3;
const int Ki_s=1,Ki_m=2,Ki_l=3;
const int Kd_s=1,Kd_m=2,Kd_l=3;
///////////
///////

/* USER CODE END PV */

/* Private function prototypes
-----
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */

```

```

//////////



void init() {
    int i,j;
    NE=0; PDE=0; NDE=0; PE=0; ZE=0; ZDE=0;
    Kp=0; Ki=0; Kd=0;
    error=0;
    prev_err=0;
    PWM_Fuzzy=0;
    err_sum=0;
    for(i=0;i<3;i++) {
        for(j=0;j<3;j++) Fuzzy_Matrix[i][j]=0;
    }
}

int max(int x,int y) {
    return x>y?x:y;
}

int min(int x,int y) {
    return x>y?y:x;
}

void Fuzzification(){
    NE=0; PDE=0; NDE=0; PE=0;
    if(error<=-20) NE=20;
    else if(error>=20) PE=20;
    else{
        if(error>=0) {
            PE=error;
            NDE=20-error;
        }
        else{
            NE=-error;
            ZDE=20+error;
        }
    }
    if(d_err<=-20) NDE=20;
    else if(d_err>=20) PDE=20;
    else{
        if(d_err>=0) {
            PDE=d_err;
            ZDE=20-d_err;
        }
    }
}

```

```

    }

    else{
        NDE=-d_err;
        ZDE=20+d_err;
    }
}

}

void Create_Fuzzy_Matrix(){
    Fuzzy_Matrix[0][0]=min(NE,NDE);
    Fuzzy_Matrix[0][1]=min(NE,ZDE);
    Fuzzy_Matrix[0][2]=min(NE,PDE);
    Fuzzy_Matrix[1][0]=min(ZE,NDE);
    Fuzzy_Matrix[1][1]=min(ZE,ZDE);
    Fuzzy_Matrix[1][2]=min(ZE,PDE);
    Fuzzy_Matrix[2][0]=min(PE,NDE);
    Fuzzy_Matrix[2][1]=min(PE,ZDE);
    Fuzzy_Matrix[2][2]=min(PE,PDE);
}

void Defuzzification(){
    Kp_small=max(Fuzzy_Matrix[0][0],Fuzzy_Matrix[0][1]);
    Kp_large=max(Fuzzy_Matrix[2][1],Fuzzy_Matrix[2][2]);

    Kp_medium=max(Fuzzy_Matrix[0][1],max(Fuzzy_Matrix[1][0],max(Fuzzy_Matrix[1][1],max(Fuzzy_Matrix[1][2],Fuzzy_Matrix[2][1]))));

    Ki_small=max(Fuzzy_Matrix[0][0],Fuzzy_Matrix[0][1]);
    Ki_large=max(Fuzzy_Matrix[2][1],Fuzzy_Matrix[2][2]);

    Ki_medium=max(Fuzzy_Matrix[0][1],max(Fuzzy_Matrix[1][0],max(Fuzzy_Matrix[1][1],max(Fuzzy_Matrix[1][2],Fuzzy_Matrix[2][1]))));

    Kd_large=max(Fuzzy_Matrix[0][0],Fuzzy_Matrix[0][1]);
    Kd_small=max(Fuzzy_Matrix[2][1],Fuzzy_Matrix[2][2]);

    Kd_medium=max(Fuzzy_Matrix[0][1],max(Fuzzy_Matrix[1][0],max(Fuzzy_Matrix[1][1],max(Fuzzy_Matrix[1][2],Fuzzy_Matrix[2][1]))));

    if(Kp_small!=0 || Kp_medium!=0 || Kp_large!=0)
        Kp=(Kp_s*Kp_small+Kp_m*Kp_medium+Kp_l*Kp_large)/(Kp_small+Kp_medium+Kp_large);
}

```

```

if(Ki_small!=0 || Ki_medium!=0 || Ki_large!=0)
Ki=(Ki_s*Ki_small+Ki_m*Ki_medium+Ki_l*Ki_large)/(Ki_small+Ki_medium+Ki_
large);
if(Kd_small!=0 || Kd_medium!=0 || Kd_large!=0)
Kd=(Kd_s*Kd_small+Kd_m*Kd_medium+Kd_l*Kd_large)/(Kd_small+Kd_medium+Kd_
large);
}

int Compute_PWM() {
err_sum+=error;
out=Kp*error + Ki*err_sum + Kd*d_err;
if(out>127) out=127;
else if(out<0) out=0;
prev_err=error;
return out;
}

//int PWM_Fuzzy = 200;

void Input_Fuzzy(int input)
{

//Receiving Data as ticks
ticks=input;
//Call the Fuzzy Code Function Here and return the value to PWM_Fuzzy
error=target-ticks;
d_err=error-prev_err;
Fuzzification();
Create_Fuzzy_Matrix();
Defuzzification();
PWM_Fuzzy=Compute_PWM();

}
///////////
///////
/* USER CODE END PFP */

/* Private user code
-----*/
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

```

```

/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /* MCU
Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the
Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

    /* USER CODE END SysInit */

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */

    /////////////////
////

//init();
//int input_ = 60;
//Send data
//Input_Fuzzy(input_);

```

```

//////////



/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
   



//Send PWM;





/* USER CODE END WHILE */

/* USER CODE BEGIN 3 */
}

/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

    /** Configure the main internal regulator output voltage
    */
    if (HAL_PWREx_ControlVoltageScaling(PWR_REGULATOR_VOLTAGE_SCALE1) != HAL_OK)
    {
        Error_Handler();
    }
}

```

```

/** Configure LSE Drive Capability
*/
HAL_PWR_EnableBkUpAccess();
__HAL_RCC_LSEDRIVE_CONFIG(RCC_LSEDRIVE_LOW);

/** Initializes the RCC Oscillators according to the specified
parameters
* in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType =
RCC_OSCILLATORTYPE_LSE|RCC_OSCILLATORTYPE_MSI;
RCC_OscInitStruct.LSEState = RCC_LSE_ON;
RCC_OscInitStruct.MSISState = RCC_MSI_ON;
RCC_OscInitStruct.MSICalibrationValue = 0;
RCC_OscInitStruct.MSIClockRange = RCC_MSIRANGE_6;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_MSI;
RCC_OscInitStruct.PLL.PLLM = 1;
RCC_OscInitStruct.PLL.PLLN = 16;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV7;
RCC_OscInitStruct.PLL.PLLQ = RCC_PLLQ_DIV2;
RCC_OscInitStruct.PLL.PLLR = RCC_PLLR_DIV2;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                            |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
{
    Error_Handler();
}

/** Enable MSI Auto calibration

```

```

        */
    HAL_RCCEx_EnableMSIPLLMode();
}

/***
 * @brief USART2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART2_UART_Init(void)
{
    /* USER CODE BEGIN USART2_Init_0 */

    /* USER CODE END USART2_Init_0 */

    /* USER CODE BEGIN USART2_Init_1 */

    /* USER CODE END USART2_Init_1 */
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    huart2.Init.OneBitSampling = UART_ONE_BIT_SAMPLE_DISABLE;
    huart2.AdvancedInit.AdvFeatureInit = UART_ADVFEATURE_NO_INIT;
    if (HAL_UART_Init(&huart2) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN USART2_Init_2 */

    /* USER CODE END USART2_Init_2 */
}

/***
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
*/

```

```

*/
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(OUTPUT_GPIO_Port, OUTPUT_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin : INPUT_Pin */
GPIO_InitStruct.Pin = INPUT_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(INPUT_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : OUTPUT_Pin */
GPIO_InitStruct.Pin = OUTPUT_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(OUTPUT_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : LD3_Pin */
GPIO_InitStruct.Pin = LD3_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(LD3_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

```

```

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return
     state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifndef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line
 * number
 *          where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and
     line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
     line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```

F446RE_LCD

main.c

```

/* USER CODE BEGIN Header */
/**



*****
* @file          : main.c
* @brief         : Main program body

*****



*****
* @attention
*
* Copyright (c) 2024 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the
LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.
*



*****



*/
/* USER CODE END Header */                                     Includes
-----*/



#include "main.h"



/*                                         Private                               includes
-----*/



/* USER CODE BEGIN Includes */



//////////



#include <stdlib.h>
#include <stdio.h>
#include "lcd.h"
#include "Fuzzy.h"
#include "max31856.h"
//////////
//////////




```

```

/* USER CODE END Includes */

/*
-----*----- Private -----*----- typedef
-----*-----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/*
-----*----- Private -----*----- define
-----*-----*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/*
-----*----- Private -----*----- macro
-----*-----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/*
-----*----- Private -----*----- variables
-----*-----*/
SPI_HandleTypeDef hspi2;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

///////////////////////////////
//Definimos conjunto para variable de entrada
float USAL[] = {-50,50}; // Definimos el universo de discurso (rango) de la variable salida
float UIN[] = {-1500,1500}// Definimos universo de discurso de la variable de entrada
float ENP[] = {-1500,-666,0,'T'}; //Definimos los conjuntos difusos, colocando 'T' si es triangular, o 'R' si es trapezoidal
float EC[] = {-5,0,5,'T'};// Conjuntos difusos de entrada
float EPP[] = {6,666,1500,'T'};

// Definimos conjuntos variable de salida
float DT[] = {-40,-30,-20,-10,'R'}; //Conjunto difuso trapezoidal
float ZE[] = {-10,0,10,'T'}; //Conjunto difuso triangular

```

```

float AT[] = {10,20,30,40,'R'};// Conjunto difuso trapezoidal
float paso = 2; //Se define la cantidad mínima de variación para los
numeros del conjunto de salida
//OJO: La entrada no debe llegar a los valores extremos de su conjunto
float setpoint =1000,rpm=0,error;
int tam;
///////////
///////



/* USER CODE END PV */



/*
----- Private function prototypes -----
*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_SPI2_Init(void);
/* USER CODE BEGIN PFP */



/* USER CODE END PFP */



/*
----- Private user code -----
*/
/* USER CODE BEGIN 0 */



/* USER CODE END 0 */



/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void)
{
/* USER CODE BEGIN 1 */



    tam = calc_size(USAL,paso); // Calculamos el tamaño que tendrá el
vector del conjunto de salida con universo salida y "paso"

/* USER CODE END 1 */



/*
----- MCU Configuration -----
*/

```

```

/* Reset of all peripherals, Initializes the Flash interface and the
Systick.*/
HAL_Init();

/* USER CODE BEGIN Init */

/* USER CODE END Init */

/* Configure the system clock */
SystemClock_Config();

/* USER CODE BEGIN SysInit */

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
MX_SPI2_Init();
/* USER CODE BEGIN 2 */

/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////
max31856_t therm = {&hspi2, {GPIOC, GPIO_PIN_3}};
max31856_init(&therm);
max31856_set_noise_filter(&therm, CR0_FILTER_OUT_50Hz);
max31856_set_cold_junction_enable(&therm, CR0_CJ_ENABLED);
max31856_set_thermocouple_type(&therm, CR1_TC_TYPE_K);
max31856_set_average_samples(&therm, CR1_AVG_TC_SAMPLES_2);
    max31856_set_open_circuit_fault_detection(&therm,
CR0_OC_DETECT_ENABLED_TC_LESS_2ms);
max31856_set_conversion_mode(&therm, CR0_CONV_CONTINUOUS);

/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////


/////////////////////////////////////////////////////////////////
/////////////////////////////////////////////////////////////////
Lcd_PortType ports[] = { GPIOC, GPIOB, GPIOA, GPIOA };
    Lcd_PinType pins[] = {GPIO_PIN_7, GPIO_PIN_6, GPIO_PIN_7,
GPIO_PIN_6};

```

```

Lcd_HandleTypeDef lcd;
lcd = Lcd_create(ports, pins, GPIOB, GPIO_PIN_5, GPIOB, GPIO_PIN_4,
LCD_4_BIT_MODE);
/*
Lcd_clear(&lcd);
Lcd_cursor(&lcd, 0,0);
Lcd_string(&lcd, "Input: ");
Lcd_cursor(&lcd, 1,0);
Lcd_string(&lcd, "Output: */

Lcd_clear(&lcd);
Lcd_cursor(&lcd, 0,0);
Lcd_string(&lcd, "Temp: ");

///////////
///////////

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    float B[tam]; //Creamos el vector que guardará los numeros del
conjunto de salida
    inicio(B,tam); //Inicializamos el objeto fuzzy
    error=setpoint-rpm;//Calculamos la variable error
    /*
     * Se empieza con la inferencia, y para ello las reglas. Para llamar
a una regla con una entrada se utiliza la función "regla_simple"
     * la cual recibe como parametros (Conjunto entrada,Universo de
entrada,variable a evaluar,conjunto salida,universo salida,vector
salida,tamaño
     * vector salida)
    */
    regla_simple(END,UIN,error,DT,USAL,B,tam);//Llamamos a la función
para aplicar la regla correspondiente
    regla_simple(EC,UIN,error,ZE,USAL,B,tam);
    regla_simple(EPP,UIN,error,AT,USAL,B,tam);
    float res = defusi(B,USAL,tam); //Se llama a la función para
defusificar el conjunto de salida y obtener el resultado del sistema
fuzzy
}

```

```

rpm = rpm+res; // Se asigna ese resultado a la variable control.

*****
/*
Lcd_clear(&lcd);
Lcd_cursor(&lcd, 0,0);
Lcd_string(&lcd, "Input: ");
Lcd_cursor(&lcd, 0,7);
Lcd_int(&lcd, setpoint);
Lcd_cursor(&lcd, 1,0);
Lcd_string(&lcd, "Output: ");
Lcd_cursor(&lcd, 1,8);
Lcd_int(&lcd, rpm);
*/
*****
//Se imprime el valor de rpm que debería converger al valor de
setpoint
//HAL_Delay(500);

*****
max31856_read_fault(&therm);

if (therm.sr.val) {
    /* Handle thermocouple error */
}
float temp = max31856_read_TC_temp(&therm);

*****
Lcd_clear(&lcd);
Lcd_cursor(&lcd, 0,0);
Lcd_string(&lcd, "Temp: ");
Lcd_cursor(&lcd, 0,7);
Lcd_int(&lcd, (int)temp);

HAL_Delay(500);
//printf("%d", (int)temp);
//fflush(stdout);

*****

```

```
//////////  
/////////  
    //Send PWM;  
  
//////////  
/////////  
  
/* USER CODE END WHILE */  
  
/* USER CODE BEGIN 3 */  
  
}  
/* USER CODE END 3 */  
}  
  
/**  
 * @brief System Clock Configuration  
 * @retval None  
 */  
void SystemClock_Config(void)  
{  
    RCC_OscInitTypeDef RCC_OscInitStruct = {0};  
    RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};  
  
    /** Configure the main internal regulator output voltage  
    */  
    __HAL_RCC_PWR_CLK_ENABLE();  
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);  
  
    /** Initializes the RCC Oscillators according to the specified  
    parameters  
     * in the RCC_OscInitTypeDef structure.  
    */  
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;  
    RCC_OscInitStruct.HSISState = RCC_HSI_ON;  
    RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;  
    RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;  
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;  
    RCC_OscInitStruct.PLL.PLLM = 16;  
    RCC_OscInitStruct.PLL.PLLN = 336;  
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;  
    RCC_OscInitStruct.PLL.PLLQ = 2;
```

```

RCC_OscInitStruct.PLL.PLLR = 2;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
 */
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                            |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) != HAL_OK)
{
    Error_Handler();
}

/**
 * @brief SPI2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_SPI2_Init(void)
{

/* USER CODE BEGIN SPI2_Init_0 */

/* USER CODE END SPI2_Init_0 */

/* USER CODE BEGIN SPI2_Init_1 */

/* USER CODE END SPI2_Init_1 */

/* SPI2 parameter configuration*/
hspi2.Instance = SPI2;
hspi2.Init.Mode = SPI_MODE_MASTER;
hspi2.Init.Direction = SPI_DIRECTION_2LINES;
hspi2.Init.DataSize = SPI_DATASIZE_8BIT;
hspi2.Init.CLKPolarity = SPI_POLARITY_LOW;

```

```

hspi2.Init.CLKPhase = SPI_PHASE_1EDGE;
hspi2.Init.NSS = SPI_NSS_SOFT;
hspi2.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_2;
hspi2.Init.FirstBit = SPI_FIRSTBIT_MSB;
hspi2.Init.TIMode = SPI_TIMODE_DISABLE;
hspi2.Init.CRCCalculation = SPI_CRCALCULATION_DISABLE;
hspi2.Init.CRCPolynomial = 10;
if (HAL_SPI_Init(&hspi2) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN SPI2_Init 2 */

/* USER CODE END SPI2_Init 2 */

}

/**
 * @brief USART2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART2_UART_Init(void)
{

/* USER CODE BEGIN USART2_Init 0 */

/* USER CODE END USART2_Init 0 */

/* USER CODE BEGIN USART2_Init 1 */

/* USER CODE END USART2_Init 1 */
huart2.Instance = USART2;
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart2) != HAL_OK)
{
    Error_Handler();
}

```

```

}

/* USER CODE BEGIN USART2_Init_2 */

/* USER CODE END USART2_Init_2 */

}

/***
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOC, CS_Pin|GPIO_PIN_7, GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOA, LD2_Pin|GPIO_PIN_6|GPIO_PIN_7,
GPIO_PIN_RESET);

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(GPIOB, GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6,
GPIO_PIN_RESET);

    /*Configure GPIO pin : B1_Pin */
    GPIO_InitStruct.Pin = B1_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

    /*Configure GPIO pins : CS_Pin PC7 */
    GPIO_InitStruct.Pin = CS_Pin|GPIO_PIN_7;

```

```

GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pins : LD2_Pin PA6 PA7 */
GPIO_InitStruct.Pin = LD2_Pin|GPIO_PIN_6|GPIO_PIN_7;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pins : PB4 PB5 PB6 */
GPIO_InitStruct.Pin = GPIO_PIN_4|GPIO_PIN_5|GPIO_PIN_6;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

```

```

#ifndef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line
number
 *
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
/* USER CODE BEGIN 6 */
/* User can add his own implementation to report the file name and
line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
/* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```

F446RE_FuzzyController

main.c

```

/* USER CODE BEGIN Header */
/**


*****
* @file          : main.c
* @brief         : Main program body

*****


*****
* @attention
*
* Copyright (c) 2024 STMicroelectronics.
* All rights reserved.
*
* This software is licensed under terms that can be found in the
LICENSE file
* in the root directory of this software component.
* If no LICENSE file comes with this software, it is provided AS-IS.

```

```

/*
***** *****
**** */

/* USER CODE END Header */

/*
                                         Includes
-----* */

#include "main.h"

/*
                                         Private
-----* */

/* USER CODE BEGIN Includes */
#include <stdlib.h>
#include <stdio.h>
#include "lcd.h"
/* USER CODE END Includes */

/*
                                         Private
-----* */

/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/*
                                         Private
-----* */

/* USER CODE BEGIN PD */

/* USER CODE END PD */

/*
                                         Private
-----* */

/* USER CODE BEGIN PM */

/* USER CODE END PM */

/*
                                         Private
-----* */

UART_HandleTypeDef huart4;
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

```

```

/* USER CODE END PV */

/*
----- Private function prototypes -----
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_UART4_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/*
----- Private user code -----
/* USER CODE BEGIN 0 */

/* USER CODE END 0 */

/**
 * @brief  The application entry point.
 * @retval int
 */
int main(void)
{
    /* USER CODE BEGIN 1 */

    /* USER CODE END 1 */

    /*
----- Configuration -----
    */

    /* Reset of all peripherals, Initializes the Flash interface and the
    Systick. */
    HAL_Init();

    /* USER CODE BEGIN Init */

    /* USER CODE END Init */

    /* Configure the system clock */
    SystemClock_Config();

    /* USER CODE BEGIN SysInit */

```

```

/* USER CODE END SysInit */

/* Initialize all configured peripherals */
MX_GPIO_Init();
MX_USART2_UART_Init();
MX_UART4_Init();
/* USER CODE BEGIN 2 */

// Lcd_PortType ports[] = { D4_GPIO_Port, D5_GPIO_Port, D6_GPIO_Port,
D7_GPIO_Port };
Lcd_PortType ports[] = {GPIOC, GPIOB, GPIOA, GPIOA};
// Lcd_PinType pins[] = {D4_Pin, D5_Pin, D6_Pin, D7_Pin};
Lcd_PinType pins[] = {GPIO_PIN_7, GPIO_PIN_6, GPIO_PIN_7, GPIO_PIN_6};
Lcd_HandleTypeDef lcd;
// Lcd_create(ports, pins, RS_GPIO_Port, RS_Pin, EN_GPIO_Port, EN_Pin,
LCD_4_BIT_MODE);
lcd = Lcd_create(ports, pins, GPIOB, GPIO_PIN_5, GPIOB, GPIO_PIN_4,
LCD_4_BIT_MODE);
Lcd_cursor(&lcd, 0, 0);
Lcd_clear(&lcd);

/* USER CODE END 2 */
Lcd_clear(&lcd);
Lcd_int(&lcd, 15);
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
}
/* USER CODE END 3 */
}

/**
 * @brief System Clock Configuration
 * @retval None
 */
void SystemClock_Config(void)
{

```

```

RCC_OscInitTypeDef RCC_OscInitStruct = {0};
RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

/** Configure the main internal regulator output voltage
*/
__HAL_RCC_PWR_CLK_ENABLE();
__HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE3);

/** Initializes the RCC Oscillators according to the specified
parameters
* in the RCC_OscInitTypeDef structure.
*/
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSISState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
RCC_OscInitStruct.PLL.PLLM = 16;
RCC_OscInitStruct.PLL.PLLN = 336;
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
RCC_OscInitStruct.PLL.PLLQ = 2;
RCC_OscInitStruct.PLL.PLLR = 2;
if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
{
    Error_Handler();
}

/** Initializes the CPU, AHB and APB buses clocks
*/
RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                            |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_2) !=
HAL_OK)
{
    Error_Handler();
}

```

```

/**
 * @brief UART4 Initialization Function
 * @param None
 * @retval None
 */
static void MX_UART4_Init(void)
{

/* USER CODE BEGIN UART4_Init_0 */

/* USER CODE END UART4_Init_0 */

/* USER CODE BEGIN UART4_Init_1 */

/* USER CODE END UART4_Init_1 */
huart4.Instance = UART4;
huart4.Init.BaudRate = 115200;
huart4.Init.WordLength = UART_WORDLENGTH_8B;
huart4.Init.StopBits = UART_STOPBITS_1;
huart4.Init.Parity = UART_PARITY_NONE;
huart4.Init.Mode = UART_MODE_TX_RX;
huart4.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart4.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart4) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN UART4_Init_2 */

/* USER CODE END UART4_Init_2 */

}

/**
 * @brief USART2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART2_UART_Init(void)
{

/* USER CODE BEGIN USART2_Init_0 */

```

```

/* USER CODE END USART2_Init_0 */

/* USER CODE BEGIN USART2_Init_1 */

/* USER CODE END USART2_Init_1 */
huart2.Instance = USART2;
huart2.Init.BaudRate = 115200;
huart2.Init.WordLength = UART_WORDLENGTH_8B;
huart2.Init.StopBits = UART_STOPBITS_1;
huart2.Init.Parity = UART_PARITY_NONE;
huart2.Init.Mode = UART_MODE_TX_RX;
huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart2.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart2) != HAL_OK)
{
    Error_Handler();
}

/* USER CODE BEGIN USART2_Init_2 */

/* USER CODE END USART2_Init_2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
/* USER CODE BEGIN MX_GPIO_Init_1 */
/* USER CODE END MX_GPIO_Init_1 */

/* GPIO Ports Clock Enable */
__HAL_RCC_GPIOC_CLK_ENABLE();
__HAL_RCC_GPIOH_CLK_ENABLE();
__HAL_RCC_GPIOA_CLK_ENABLE();
__HAL_RCC_GPIOB_CLK_ENABLE();

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

```

```

/*Configure GPIO pin : B1_Pin */
GPIO_InitStruct.Pin = B1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : LD2_Pin */
GPIO_InitStruct.Pin = LD2_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return
     state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifndef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line
 * number
 *           where the assert_param error has occurred.
 * @param   file: pointer to the source file name

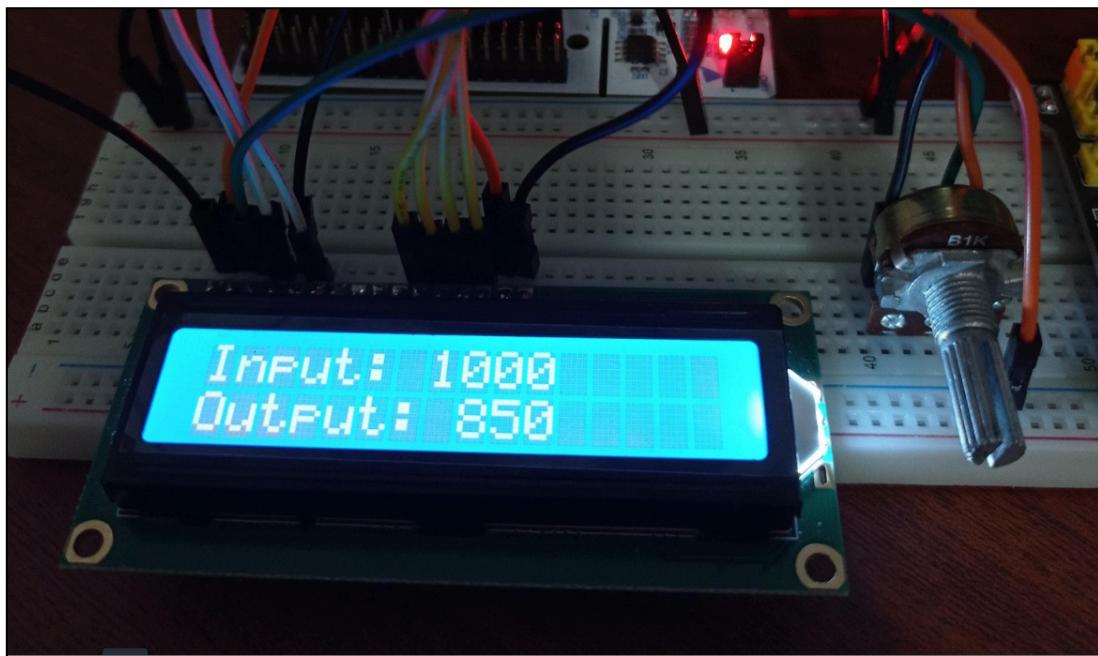
```

```

* @param line: assert_param error line source number
* @retval None
*/
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and
line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file,
line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```

Figura 14. Pruebas del programa y sistema desarrollado



IV. CONTROLADOR ON-OFF DE TEMPERATURA

Finalmente, se optó por un control On-Off, dado que se decidió desarrollar el sistema utilizando una tarjeta ESP32. Este controlador fue elegido por su eficiencia y porque no requiere algoritmos demasiado complejos. Además, de acuerdo con los resultados obtenidos, se lograron los objetivos planteados en la primera parte.

Figura 15. Circuito de referencia para la tarjeta de circuito impreso

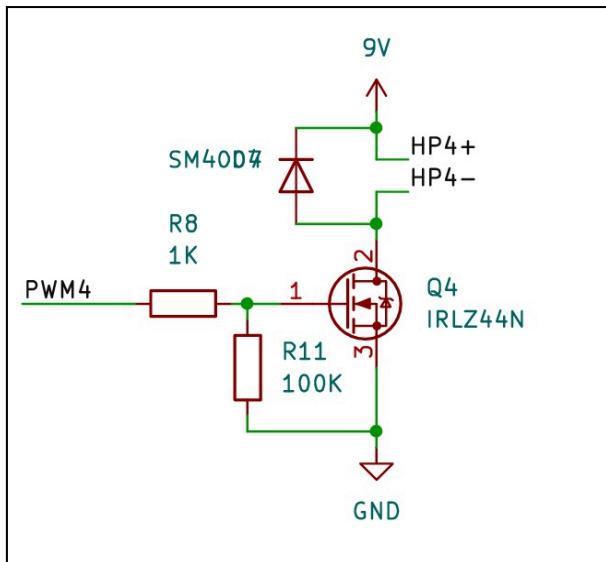


Figura 16. Circuito de referencia 1 para el desarrollo de las pruebas de control On-Off

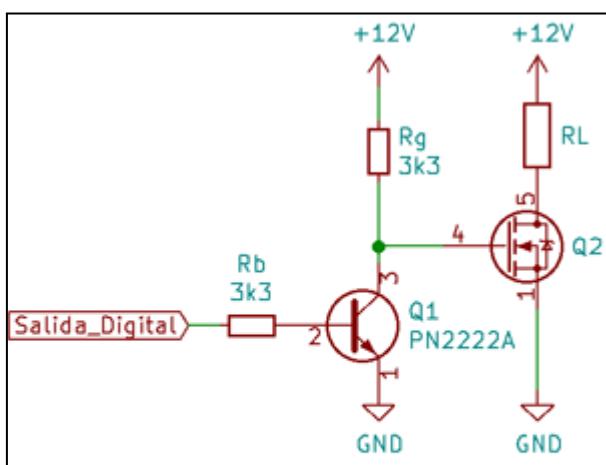


Figura 17. Circuito de referencia 2 para el desarrollo de las pruebas de control On-Off

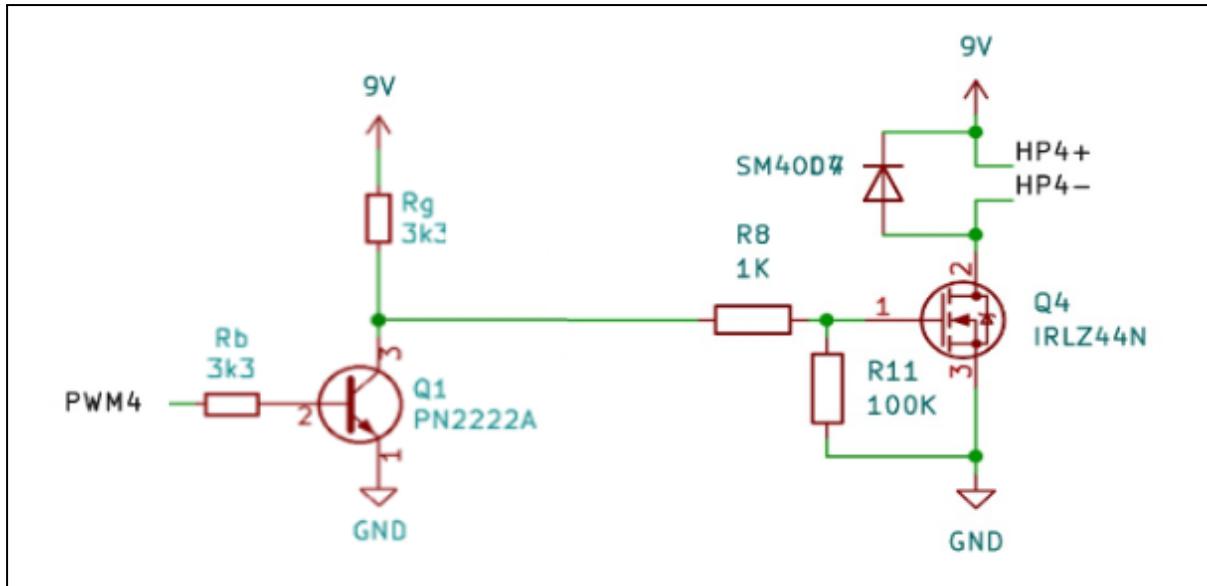
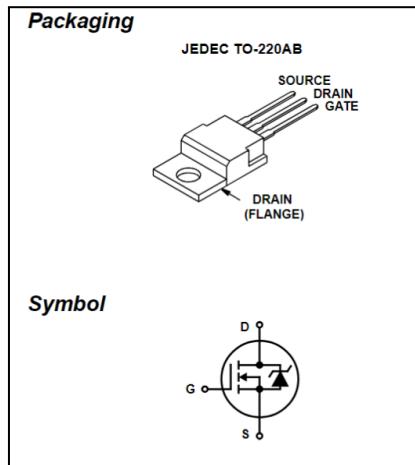


Figura 18. Integrado utilizado para las pruebas del circuito



En la Figura 18 se presenta el circuito del transistor utilizado para las pruebas finales, específicamente el modelo de MOSFET IRF540N. Según el datasheet, este transistor debe recibir un voltaje de alimentación de +9V, lo cual se tendrá en cuenta durante las pruebas. La variación del voltaje de salida se ajustará en función de una señal PWM de acuerdo a la Figura 17 y Tabla 2.

Tabla 2. Parámetros de señales para el controlador On-Off

Señal de entrada	+9V DC
Señal de salida	PWM

En la Tabla 2 se puede observar que el voltaje de entrada y de operación es de +9V. Este voltaje fue seleccionado en función de los cálculos realizados, teniendo en cuenta el consumo de corriente de los *heating pads*, considerando la corriente

máxima de cada uno. Dado que se utilizarán 6 *heating pads* dentro del payload, el voltaje adecuado y más eficiente para la alimentación resultó ser de +9V DC (ver Figura 19).

Figura 19. Consumo de corriente para cada voltaje de alimentación.

Heating pads Test				
Voltage (V)	12	9	7.5	5
Current (A)	1.402	1.162	1	0.747
Measured temperature (°C)	92.8	67.6	56.4	41.2
Ambient temperature (°C)	23.8	23.8	23.2	24.1
Temperature variation	69	43.8	33.2	17.1
Power	16.824	10.458	7.5	3.735
Total power (to 6 Heating Pads on)	100.944	62.748	45	22.41
		12.252		

Figura 20. Circuito empleado para verificar la temperatura del MOSFET.



Como se puede observar en la Figura 20, el circuito empleado consta de un ESP32 que servirá de control y será conectado al circuito propuesto. Se utiliza el MOSFET para medir la temperatura a partir de la termocupla.

ANEXOS

Anexo 1. Control On-Off

```
// Incluimos librería
#include <DHT.h>
#include <LiquidCrystal.h>

#define DHTPIN 2 // Definimos el pin digital donde se conecta el sensor
#define DHTTYPE DHT11 // Dependiendo del tipo de sensor

DHT dht(DHTPIN, DHTTYPE); // Inicializamos el sensor DHT11

const int rs = 8, en = 9, d4 = 10, d5 = 11, d6 = 12, d7 = 13; // Variables para la conexión del display LCD1602
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

volatile long contador = 0;
int interruptPin = 3;

double K = 60000; // constante
float n = 2; // numero de pulsos por revolución
float tiempo_T = 0; // Tiempo T
float RPM = 0; // revoluciones por minuto

float tiempo_inicial = 0; // tiempo el muestreo para la medición de las RPM
float pulsos_inicial = 0;
bool tiempo_muestra = false;
float error = 0;

int set = 20;
double signal_control;
int pwm;
float t;
float error_max;
float error_min;
```

```

/* Variables para la visualización en puerto Serial */

float time_seconds = 0;

/* Variables para la visualización en el display LCD */

String temp = "T: ";
String RPMS = "RPM";
String setp = "set: ";

void setup() {

    Serial.begin(9600); // Inicializamos comunicación serie
    dht.begin(); // Comenzamos el sensor DHT
    attachInterrupt(digitalPinToInterrupt(interruptPin), lectura_encoder,
    RISING); // Configuración interrupción
    lcd.begin(16, 2);
}

void loop() {

    delay(3000); // Retraso de lectura debido a las limitaciones del
    sensor
    t = dht.readTemperature(); // Leemos la temperatura en grados
    centígrados (por defecto)
    time_seconds = (millis()) / 1000;

    Serial.print("Tiempo en segundos: ");
    Serial.print(time_seconds);
    Serial.print(" Temperatura: ");
    Serial.println(t);

    impirmir_valores();

    error = set - t; // Error set(valor deseado) - t(temperatura leida por
    el sensor)

    if (error < 0)analogWrite(5, 255); // Enciende ventilador (ON)
    if (error > 0)analogWrite(5, 0); // Apaga ventilador (OFF)
}

```

```

float calculate_RPM() { // funciona para calcular RPM

    if (tiempo_muestra == true) {

        tiempo_inicial = millis();
        pulsos_inicial = abs(contador);
        tiempo_muestra = false;
    }

    tiempo_T = millis() - tiempo_inicial;

    if (tiempo_T >= 3000) { //

        RPM = K * (abs(contador) - pulsos_inicial) / (n * tiempo_T); // puso
final- pulso inicial
        tiempo_muestra = true; // habilita muestra
        contador = 0;
        return RPM;
    }

}

void lectura_encoder() { // funcion de la interrupción

    contador++;
}

void impirmir_valores() { //Imprime valores en el display LCD 1602

    lcd.setCursor(9, 0);
    lcd.print(setp + set);
    lcd.setCursor(0, 0);
    lcd.print(temp + t);

    lcd.setCursor(7, 1);
    lcd.print("RPM:");
}

```

```
lcd.setCursor(12, 1);
lcd.print(calculate_RPM(), 3);

lcd.setCursor(0, 1);
lcd.print("U");

lcd.setCursor(2, 1);
lcd.print(error);

}
```

V. PRUEBAS FINALES

Figura 21. Tareas finales por desarrollar.

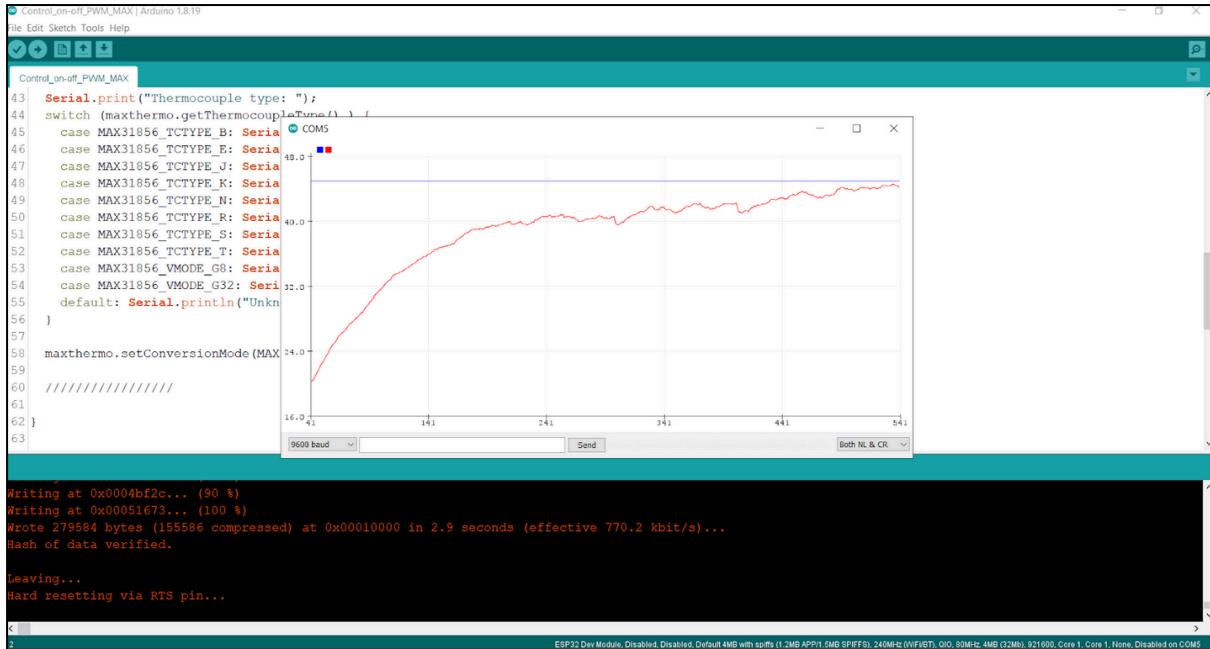
Control de temperatura (David)

1. Verificar la temperatura al 100% PWM del MOSFET
2. Validar si con menos de 100% se puede llegar al setpoint
3. Verificar a temperaturas bajas (Refrigerador o celda peltier)

Figura 22. Sistema de prueba para el control On-Off



Figura 23. Velocidad del controlador



En la Figura 23 se muestra la velocidad alcanzada para llegar a la temperatura establecida de 25 °C como set-point. El tiempo requerido para alcanzar esta temperatura fue de aproximadamente 2 minutos. Aunque este tiempo puede parecer considerablemente alto, es adecuado para el sistema, ya que el aumento de la temperatura implica un proceso lento.

Figura 24. Tabla de parámetros utilizados en el sistema para las pruebas de control PWM

Vin	9V	
PWM 4	3.3V	DESACTIVADO
PWM 4	0V	ACTIVADO
PWM	100%	
PWM 4	0V	0
Temperatura MOSFET	39.8 °C	
PWM	50%	
PWM 4	1.57V	128
Temperatura MOSFET	31.8 °C	

En la Figura 24 se observa el voltaje de entrada establecido en +9V DC, junto con el control de los pines PWM, que permiten alcanzar voltajes de 0V y 3.3V. Además, se

muestran las temperaturas alcanzadas para una variación del voltaje PWM entre el 100% y el 50% (valores de 0 a 128, respectivamente).

Control de temperatura a 25 °C con respecto a la variación del PWM.

Figura 25. Control de temperatura a 25 °C (PWM = 100%)

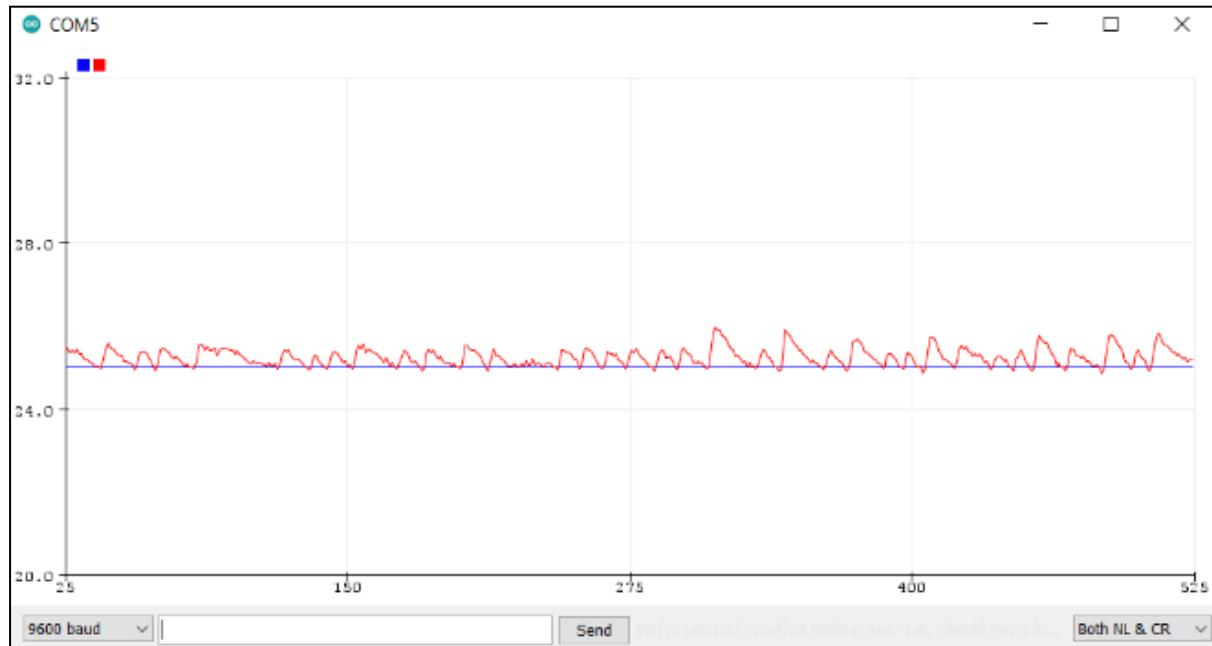


Figura 26. Control de temperatura a 25 °C (PWM = 50%)

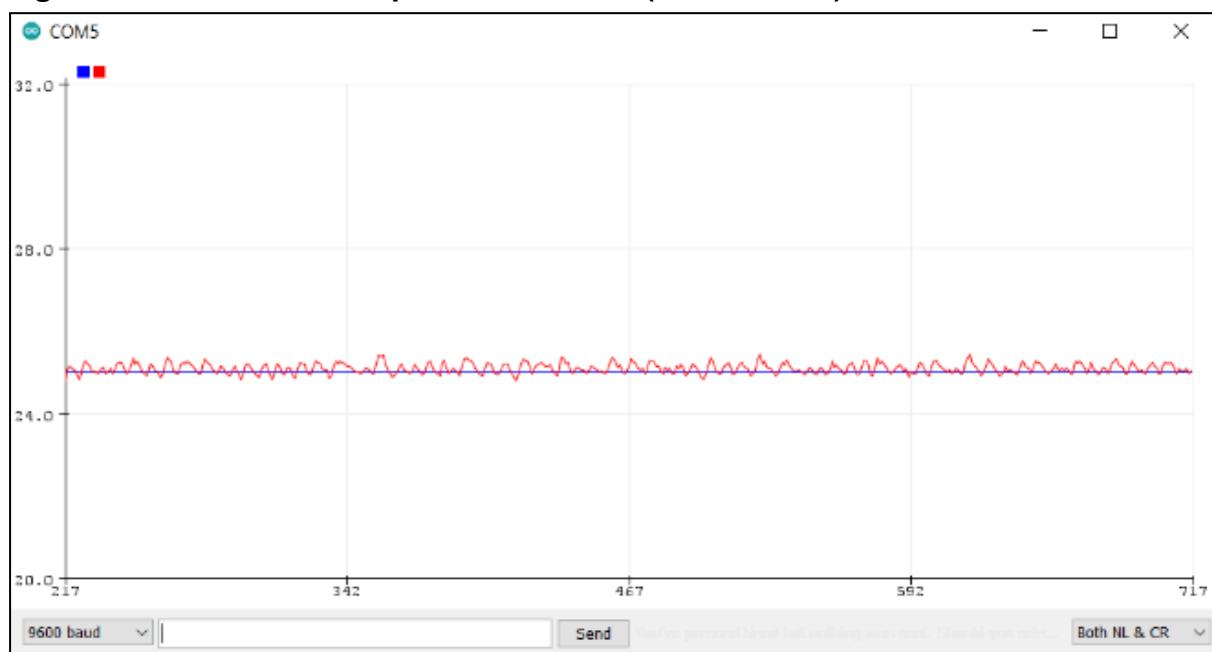
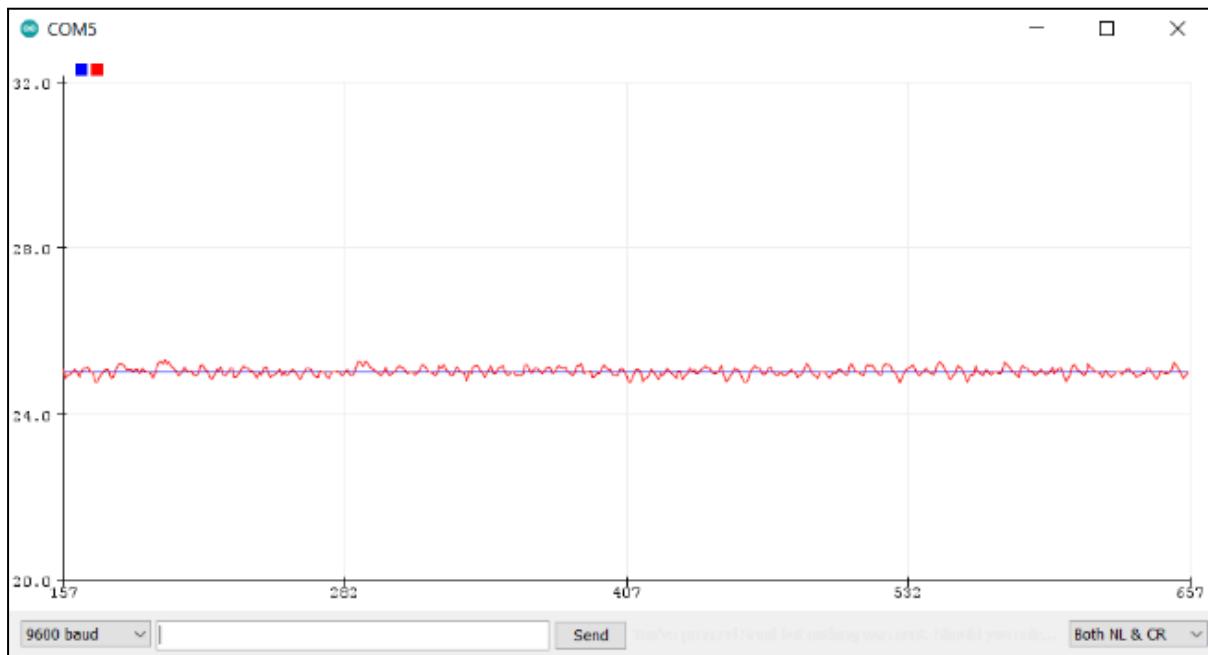


Figura 27. Control de temperatura a 25 °C (PWM = 25%)



A partir de las figuras anteriores, podemos observar que, con un valor de PWM del 25%, el controlador logra un ajuste más preciso en relación con la temperatura de referencia. A medida que el PWM aumenta, el ajuste se realiza en un rango más amplio; sin embargo, para el control de temperatura desarrollado, esto sigue siendo adecuado y no presenta ningún problema para el sistema.

Figura 28. Resultado final del control On-Off

Entrada de datos (desde Silicon Labs CP210x USB to UART Bridge (COM

Los datos procedentes del origen de datos actual aparecerán abajo según se reciban.

Datos actuales

TIME	CH1	CH2	CH3	CH4	CH5	CH6	CH7	CH8	CH9	CH10
2:09:36.43	25	25								

Información histórica										
TIME	SetPoint	Temperature	CH3	CH4	CH5	CH6	CH7	CH8	CH9	CH10
2:09:36.43	25	25								
2:09:35.44	25	25.19								
2:09:34.43	25	25.17								
2:09:33.43	25	25.23								
2:09:32.43	25	25.16								
2:09:31.44	25	25.02								
2:09:30.43	25	24.91								
2:09:29.44	25	24.97								
2:09:28.43	25	25.02								
2:09:27.43	25	25.02								
2:09:26.42	25	25.1								
2:09:25.44	25	25.04								

On-Off Temperature Control
SetPoint
Temperature

Step	SetPoint	Temperature
1	25	25.0
2	25	25.2
3	25	25.0
4	25	25.2
5	25	25.0
6	25	24.8
7	25	24.7
8	25	24.8
9	25	24.9
10	25	24.9
11	25	25.0
12	25	25.0
13	25	25.0
14	25	24.9
15	25	24.9

En la Figura 28, finalmente se muestra el control de temperatura On-Off y su funcionamiento en relación con la temperatura de referencia de 25 °C. El sistema utiliza un solo *heating pad* y una termocupla; sin embargo, puede ampliarse para incluir los 6 *heating pads* necesarios mediante el uso del protocolo SPI.

ANEXOS

Anexo 2. Control On-Off PWM

```
//***PWM MAX***/\n#include <Adafruit_MAX31856.h>\n#define DRDY_PIN 4\n\nAdafruit_MAX31856 maxthermo = Adafruit_MAX31856(5, 23, 19, 18);\n\nint pin_PWM = 16; //PWM Pin\n\n//Define PIN L298N\n//OUT1 -> (+)\n//OUT2 -> (-)\nint in1 = 32;\nint in2 = 33;\n\n///////////
```

```

int set = 25;
float t;
float error = 0;

void setup() {
    //***PWM MAX***/
    pinMode(in1, OUTPUT);
    pinMode(in2, OUTPUT);
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    pinMode(pin_PWM, OUTPUT);
    Serial.begin(9600);
    while (!Serial) delay(10);
    Serial.println("MAX31856 thermocouple test");

    pinMode(DRDY_PIN, INPUT);

    if (!maxthermo.begin()) {
        Serial.println("Could not initialize thermocouple.");
        while (1) delay(10);
    }
}

maxthermo.setThermocoupleType(MAX31856_TCTYPE_K);

Serial.print("Thermocouple type: ");
switch (maxthermo.getThermocoupleType() ) {
    case MAX31856_TCTYPE_B: Serial.println("B Type"); break;
    case MAX31856_TCTYPE_E: Serial.println("E Type"); break;
    case MAX31856_TCTYPE_J: Serial.println("J Type"); break;
    case MAX31856_TCTYPE_K: Serial.println("K Type"); break;
    case MAX31856_TCTYPE_N: Serial.println("N Type"); break;
    case MAX31856_TCTYPE_R: Serial.println("R Type"); break;
    case MAX31856_TCTYPE_S: Serial.println("S Type"); break;
    case MAX31856_TCTYPE_T: Serial.println("T Type"); break;
    case MAX31856_VMODE_G8: Serial.println("Voltage x8 Gain mode");
break;
    case MAX31856_VMODE_G32: Serial.println("Voltage x8 Gain mode");
break;
    default: Serial.println("Unknown"); break;
}

maxthermo.setConversionMode(MAX31856_CONTINUOUS);

```

```

///////////
}

void loop() {
    //***PWM MAX***//
    // The DRDY output goes low when a new conversion result is available
    int count = 0;
    while (digitalRead(DRDY_PIN)) {
        if (count++ > 200) {
            count = 0;
            Serial.print(".");
        }
    }

///////////

delay(1000);
t = maxthermo.readThermocoupleTemperature(); // Leemos la temperatura
en grados centigrados

Serial.print(set);
Serial.print(",");
Serial.print(t);
Serial.println();

//20-([-80;20])
//20-37
//set = 20
error = set - t; // Error set(valor deseado) - t(temperatura leída por
el sensor)

if (error < 0)analogWrite(pin_PWM, 0);      // Apaga Heating Pad (OFF)
if (error > 0)analogWrite(pin_PWM, 255);      // Enciende Heating Pad
(ON)
}

```

Anexo 3. Control PWM MAX

```

// This example demonstrates continuous conversion mode using the
// DRDY pin to check for conversion completion.

```

```
#include <Adafruit_MAX31856.h>
```

```

#define DRDY_PIN 4

// Use software SPI: CS, DI, DO, CLK
Adafruit_MAX31856 maxthermo = Adafruit_MAX31856(5, 23, 19, 18);
// use hardware SPI, just pass in the CS pin
//Adafruit_MAX31856 maxthermo = Adafruit_MAX31856(10);

//*****/
//Define PIN PWM
int PWM_b = 128;           //Variable 0-255 (8 bits)
int pin_PWM = 16;          //Output Pin

//Define PIN L298N
//OUT1 -> (+)
//OUT2 -> (-)
int in1 = 32;
int in2 = 33;

//PWM Characteristics
const int channel = 4;
const int frequency = 30000;
const int resolution = 8;
//*****/

void setup() {
//*****/
pinMode(in1, OUTPUT);
pinMode(in2, OUTPUT);
digitalWrite(in1,HIGH);
digitalWrite(in2,LOW);
pinMode(pin_PWM, OUTPUT);
/*
ledcSetup(channel,frequency,resolution);
ledcAttachPin(pin_PWM,channel);
*/
//*****/
Serial.begin(115200);
while (!Serial) delay(10);
Serial.println("MAX31856 thermocouple test");

pinMode(DRDY_PIN, INPUT);

if (!maxthermo.begin()) {

```

```

    Serial.println("Could not initialize thermocouple.");
    while (1) delay(10);
}

maxthermo.setThermocoupleType(MAX31856_TCTYPE_K);

Serial.print("Thermocouple type: ");
switch (maxthermo.getThermocoupleType() ) {
    case MAX31856_TCTYPE_B: Serial.println("B Type"); break;
    case MAX31856_TCTYPE_E: Serial.println("E Type"); break;
    case MAX31856_TCTYPE_J: Serial.println("J Type"); break;
    case MAX31856_TCTYPE_K: Serial.println("K Type"); break;
    case MAX31856_TCTYPE_N: Serial.println("N Type"); break;
    case MAX31856_TCTYPE_R: Serial.println("R Type"); break;
    case MAX31856_TCTYPE_S: Serial.println("S Type"); break;
    case MAX31856_TCTYPE_T: Serial.println("T Type"); break;
    case MAX31856_VMODE_G8: Serial.println("Voltage x8 Gain mode");
break;
    case MAX31856_VMODE_G32: Serial.println("Voltage x8 Gain mode");
break;
    default: Serial.println("Unknown"); break;
}

maxthermo.setConversionMode(MAX31856_CONTINUOUS);
}

void loop() {
//*****/
/*
ledcWrite(channel, PWM_b);
*/
analogWrite(pin_PWM, PWM_b);
//*****/
// The DRDY output goes low when a new conversion result is available
int count = 0;
while (digitalRead(DRDY_PIN)) {
    if (count++ > 200) {
        count = 0;
        Serial.print(".");
    }
}
Serial.println(maxthermo.readThermocoupleTemperature());
delay(10);
}

```

}

Todos los archivos desarrollados se encuentran dentro del siguiente enlace:
https://drive.google.com/drive/folders/1RA3T2mV6QtoGiJ6oQHTUSkgMli-XmbEJ?usp=drive_link