



---

**TECNOLÓGICO NACIONAL DE MÉXICO**  
INSTITUTO TECNOLÓGICO DE APIZACO

**DISEÑO E IMPLEMENTACIÓN DE SISTEMA CON CONTROL  
DIFUSO DE TEMPERATURA Y CONECTIVIDAD IOT,  
APLICADO EN SISTEMAS HIDRÁULICOS DOMÉSTICOS.**

**TITULACIÓN INTEGRAL INFORME DE RESIDENCIA  
PROFESIONAL**

**PARA OBTENER EL TÍTULO PROFESIONAL DE:  
INGENIERO ELECTRÓNICO**

**PRESENTA:  
CHRISTIAN PÉREZ MARTÍNEZ**

**NOMBRE DEL ASESOR:  
DR. CARLOS BUENO AVENDAÑO**



**APIZACO, TLAX., JUNIO 2024**



Apizaco, Tlaxcala, **03/junio/2024**  
Departamento: Ingeniería Eléctrica y Electrónica  
No. De oficio: I.E.E. /618/2024

Asunto: Liberación de Proyecto  
para la Titulación Integral

**MARTÍN ROJAS RAMÍREZ**  
**JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES**  
**PRESENTE**

Por este medio le informo que ha sido liberado el siguiente proyecto para la Titulación Integral:

|                             |                                                                                                                                      |
|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------|
| <b>Nombre del egresado:</b> | CHRISTIAN PÉREZ MARTÍNEZ                                                                                                             |
| <b>Carrera:</b>             | INGENIERÍA ELECTRÓNICA                                                                                                               |
| <b>No. De Control:</b>      | 18370629                                                                                                                             |
| <b>Nombre del proyecto:</b> | DISEÑO E IMPLEMENTACIÓN DE SISTEMA CON CONTROL DIFUSO DE TEMPERATURA Y CONECTIVIDAD IOT, APLICADO EN SISTEMAS HIDRÁULICOS DOMÉSTICOS |
| <b>Producto</b>             | TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL                                                                                |

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestros egresados.

**ATENTAMENTE**

*Excelencia en Educación Tecnológica®  
Pensar para Servir, Servir para Triunfar®*

**JAIME ALBERTO ELIZALDE OLIN**  
**JEFE DEL DEPTO. DE INGENIERÍA**  
**ELÉCTRICA Y ELECTRÓNICA**



SECRETARÍA DE EDUCACIÓN PÚBLICA  
TECNOLÓGICO NACIONAL  
DE MÉXICO  
INSTITUTO TECNOLÓGICO DE APIZACO  
DEPARTAMENTO DE  
INGENIERÍA ELÉCTRICA  
Y ELECTRÓNICA

**CARLOS BUENO AVENDAÑO**  
Nombre y firma del asesor

JAEO/Isa\*  
C.p. Archivo





**EDUCACIÓN**  
SECRETARÍA DE EDUCACIÓN PÚBLICA



TECNOLÓGICO  
NACIONAL DE MÉXICO

Instituto Tecnológico de Apizaco  
División de Estudios Profesionales

Apizaco, Tlax., **26/junio/2024**  
No. OFICIO: DEP-CT//2024

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

**CHRISTIAN PEREZ MARTINEZ**  
**EGRESADO DE LA CARRERA DE INGENIERÍA ELECTRÓNICA**  
**CON NÚMERO DE CONTROL: 18370629**  
**P R E S E N T E**

Por este medio, me permito informar a usted que por aprobación de la comisión revisora asignada para valorar el trabajo que mediante la opción **TITULACIÓN INTEGRAL INFORME DE RESIDENCIA PROFESIONAL** cuyo tema es: **"DISEÑO E IMPLEMENTACIÓN DE SISTEMA CON CONTROL DIFUSO DE TEMPERATURA Y CONECTIVIDAD IoT, APLICADO EN SISTEMAS HIDRÁULICOS DOMÉSTICOS"**, conforme a lo establecido en el procedimiento para la obtención del Título Profesional en el Instituto Tecnológico, la División de Estudios Profesionales a mi cargo le otorga:

### AUTORIZACIÓN DE IMPRESIÓN

Debiendo entregar **3** empastados del mismo, apropiadamente encuadernados, a la brevedad, para presentar su acto de recepción profesional.

Sin otro particular por el momento, le envió un cordial saludo.

### ATENTAMENTE

*Excelencia en Educación Tecnológica  
Pensar para Servir, Servir para Triunfar*



SECRETARÍA DE EDUCACIÓN PÚBLICA  
TECNOLÓGICO NACIONAL  
DE MÉXICO  
INSTITUTO TECNOLÓGICO DE APIZACO  
ESTUDIOS PROFESIONALES

**CÉSAR REYNALDO RAMOS GÓMEZ**  
**JEFE DE LA DIVISIÓN DE ESTUDIOS PROFESIONALES**

c.p. archivo



Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec, Municipio de Tzompantepec,  
Tlaxcala, Mex. C.P. 90491 Tel. (241) 4172010 Ext. 304, e-mail:  
dep\_titulacion@apizaco.tecnm.mx



**2024**  
**Felipe Carrillo**  
**PUERTO**  
MEMORIA DEL TITULACIÓN  
DEL INSTITUTO TECNOLÓGICO DE APIZACO

## **Agradecimientos.**

Doy gracias a mi familia que me han apoyado durante toda mi vida y especialmente durante mi carrera, cuando el estrés y el cansancio eran más notorios. A mi madre Yolanda que nunca me dejó solo y me ayudó a comprender que no debo rendirme tan rápido. A mis hermanos, José y Zuleyca, que siempre estuvieron conmigo. A mis amigos de la universidad que apoyaron mis proyectos y velaron constantemente por nuestros intereses en común.

también doy gracias a los profesores que me instruyeron dentro de todas las ramas de la electrónica, mostrando su comprensión y apoyo durante sus clases, especialmente a los que me guiaron y apoyaron para el desarrollo de este proyecto. Dichos profesores me apoyaron dentro y fuera del aula y considero que, si no fuera por ellos, el presente proyecto no hubiera visto la luz. Al Dr. Carlos Bueno, quien guio y asesoró este proyecto desde su conocimiento, brindándome la confianza y su disponibilidad para aclarar algunas dudas que surgieron. Al Dr Roberto Morales, por sus enseñanzas en industria 4.0, las cuales generaron curiosidad en mí, para desarrollar este proyecto y por haber encendido en mí un profundo deseo de continuar con mis estudios a nivel de posgrado. al Mtro. Alfredo Matlalcuatzi, por sus enseñanzas en programación y su asesoramiento en el desarrollo del software que utiliza este proyecto.

Finalmente, agradezco de todo corazón a todos aquellos que, de una manera u otra, han contribuido a la realización de este proyecto. Sin su apoyo, no hubiera sido posible.



# INDICE GENERAL.

|                                             |    |
|---------------------------------------------|----|
| Agradecimientos.....                        | 4  |
| Lista de Figuras.....                       | 8  |
| Capitulo I. Generalidades del proyecto..... | 13 |
| 1.1 Introducción.....                       | 13 |
| 1.2 Datos de la empresa.....                | 14 |
| 1.2.1 Área del trabajo del estudiante.....  | 17 |
| 1.3 Planteamiento del problema.....         | 19 |
| 1.4 Objetivos.....                          | 21 |
| 1.4.1 Objetivo general.....                 | 21 |
| 1.4.2 Objetivos específicos.....            | 21 |
| 1.5 Justificación.....                      | 22 |
| Capitulo II. Marco teórico.....             | 24 |
| 2.1 Definición de domótica.....             | 24 |
| 2.2 Dispositivos del IoT.....               | 26 |
| 2.3 Conexión entre dispositivos IoT.....    | 27 |
| 2.4 Aplicación IoT.....                     | 28 |
| 2.5 Interfaz de usuario.....                | 28 |
| 2.6 Lógica difusa.....                      | 29 |
| 2.7 Sensor de temperatura.....              | 30 |
| 2.7.1 Sensor de temperatura DS18B20.....    | 31 |
| 2.8 Caudalímetro.....                       | 32 |
| 2.8.1 Caudalímetro YF-S201.....             | 33 |
| 2.9 Nema 17 Motor A Pasos 17HS4401.....     | 35 |
| 2.9.1 Driver TB6600.....                    | 36 |
| 2.10 Llave de paso.....                     | 37 |
| 2.11 Raspberry Pi4.....                     | 38 |
| 2.11.1 Programación con Scratch.....        | 47 |
| 2.11.2 Programación con Python.....         | 50 |
| 2.11.3 Programación en Java.....            | 52 |
| 2.11.4 Programación en C++.....             | 53 |

|                                                                                  |     |
|----------------------------------------------------------------------------------|-----|
| 2.11.5 Programación en HTML .....                                                | 54  |
| 2.12 Pantalla táctil.....                                                        | 57  |
| 2.13 Antecedentes.....                                                           | 57  |
| 2.13.1 Diseño y desarrollo de la ducha inteligente ¡Shower!. .....               | 57  |
| 2.13.2 Rivos: Diseño y conceptualización de una ducha inteligente.....           | 58  |
| 2.13.3 Sistema de ducha termostático, sistema inteligente de chorro. ....        | 59  |
| 2.14.4 Juego de ducha inteligente, cabezal de ducha LED de techo. ....           | 60  |
| 2.13.5 Panel de ducha de acero inoxidable modelo Imperial Solana. ....           | 60  |
| 2.13.6 Sistema ducha digital baño inteligente indicador.....                     | 61  |
| Capítulo III. Desarrollo e implementación.....                                   | 63  |
| 3.1 Diseño del control difuso en el software Matlab y su análogo en Python.....  | 63  |
| 3.2 Diseño de circuito electrónico .....                                         | 78  |
| 3.3 Creación en 3D del prototipo por medio de software.....                      | 79  |
| 3.4 Programación de interfaz de usuario en el panel táctil y aplicación web..... | 82  |
| 3.5 Pruebas preliminares.....                                                    | 91  |
| 3.5.1 Prueba unitaria de integración para caudalímetro.....                      | 91  |
| 3.5.2 Pruebas unitarias de integración para motores NEMA 17 y drivers. ....      | 100 |
| 3.5.3 Pruebas unitarias de integración para sensores DS18B20.....                | 111 |
| 3.6 Etapa de implementación. ....                                                | 123 |
| 3.6.1 Dimensionamiento del prototipo físico. ....                                | 123 |
| 3.7 Creación del prototipo con el sistema hidráulico y electrónico. ....         | 126 |
| 3.7.1 Ensamblado del sistema hidráulico y electrónico. ....                      | 126 |
| Capítulo IV. Resultados .....                                                    | 145 |
| 4.1 Simulación de control difuso con Matlab.....                                 | 145 |
| 4.2 Integración de bloques y software. ....                                      | 156 |
| 4.3 Demostración del funcionamiento del sistema completo. ....                   | 163 |
| Capítulo V. Conclusiones.....                                                    | 174 |
| Competencias desarrolladas y/o aplicadas. ....                                   | 175 |
| Referencias.....                                                                 | 177 |
| Anexos.....                                                                      | 181 |
| Anexo 1. Hoja de datos del sensor de temperatura DS18B20. ....                   | 181 |
| Anexo 2. Hoja de datos del sensor de flujo YF-S201. ....                         | 209 |
| Anexo 3. Hoja de datos del motor a pasos NEMA 17. ....                           | 211 |

|                                                                                         |            |
|-----------------------------------------------------------------------------------------|------------|
| <b>Anexo 4. Hoja de datos del controlador TB6600. ....</b>                              | <b>212</b> |
| <b>Anexo 5. Hoja de datos de la Raspberry Pi 4 Modelo B.....</b>                        | <b>214</b> |
| <b>Anexo 6. Código de programación en Python del programa para la Raspberry pi.....</b> | <b>227</b> |
| <b>Anexo 7. Código en HTML para visualizar la aplicación web.....</b>                   | <b>255</b> |

## Lista de Figuras.

|                                                                                              |    |
|----------------------------------------------------------------------------------------------|----|
| Figura 1 Estructura Organizativa de Instituto Tecnológico de Apizaco. ....                   | 15 |
| Figura 2 Organigrama de Instituto Tecnológico de Apizaco .....                               | 15 |
| Figura 3 Ubicación de Instituto Tecnológico de Apizaco. ....                                 | 16 |
| Figura 4 Ubicación de Instituto Tecnológico de Apizaco – Posgrado e Investigación.....       | 16 |
| Figura 5 Aula alfa 17 Posgrado e investigación.....                                          | 17 |
| Figura 6 Área donde se desarrollaron las pruebas preliminares. ....                          | 18 |
| Figura 7 Esquema de una vivienda domótica.....                                               | 24 |
| Figura 8 Actuadores y sensores. ....                                                         | 26 |
| Figura 9 Dispositivos IoT. ....                                                              | 27 |
| Figura 10 Conexión entre dispositivos IoT. ....                                              | 28 |
| Figura 11 Sensor de temperatura DS18B20.....                                                 | 32 |
| Figura 12 Caudalímetro YF-S201. ....                                                         | 34 |
| Figura 13 Nema 17 Motor a pasos 17HS4401 .....                                               | 36 |
| Figura 14 Driver TB6600 .....                                                                | 37 |
| Figura 15 Raspberry Pi.....                                                                  | 39 |
| Figura 16 Puerto de alimentación. ....                                                       | 40 |
| Figura 17 Puerto de visualización DSI .....                                                  | 40 |
| Figura 18 Inalámbrico/Bluetooth .....                                                        | 41 |
| Figura 19 Micro HDMI 0 .....                                                                 | 41 |
| Figura 20 Sistema en chip. ....                                                              | 42 |
| Figura 21 GPIO .....                                                                         | 42 |
| Figura 22 RAM.....                                                                           | 43 |
| Figura 23 CSIPOF.....                                                                        | 43 |
| Figura 24 Av 3.5 mm .....                                                                    | 44 |
| Figura 25 Circuito integrado de administración de energía (PMIC). ....                       | 44 |
| Figura 26 Puerto USB 2.0.....                                                                | 45 |
| Figura 27 Puerto Ethernet.....                                                               | 45 |
| Figura 28 MicroSD.....                                                                       | 46 |
| Figura 29 Interfaz Scratch. ....                                                             | 48 |
| Figura 30 Ventana de Scratch .....                                                           | 49 |
| Figura 31 IDE de Python. ....                                                                | 50 |
| Figura 32 Programa ejecutado.....                                                            | 52 |
| Figura 33 Logo de C++ .....                                                                  | 54 |
| Figura 34 Primera línea de HTML.....                                                         | 56 |
| Figura 35 Ventana de comandos de Matlab con el comando fuzzy. ....                           | 63 |
| Figura 36 Inicio de la ventana del diseño de lógica difusa. ....                             | 64 |
| Figura 37 Asignación de variables a utilizar.....                                            | 65 |
| Figura 38 Visualización de la función de pertenencia de la variable error. ....              | 65 |
| Figura 39 Visualización de las funciones de pertenencia para las variables cali y fria. .... | 66 |
| Figura 40 Reglas establecidas en Matlab.....                                                 | 67 |
| Figura 41 Exportación de archivo al espacio de trabajo (Workspace).....                      | 68 |
| Figura 42 Workspace de Matlab con el archivo de control difuso. ....                         | 68 |



|                                                                                               |    |
|-----------------------------------------------------------------------------------------------|----|
| Figura 43 Exportación de archivo como documento de extensión .fis. ....                       | 69 |
| Figura 44 carpeta de guardado del archivo de control difuso.....                              | 69 |
| Figura 45 Cambio en el directorio de la ventana de inicio de Matlab. ....                     | 70 |
| Figura 46 Esquema de Simulink para la simulación del control difuso.....                      | 70 |
| Figura 47 Asignación del archivo con extensión. fis al bloque de lógica difusa. ....          | 71 |
| Figura 48 Simulación del control difuso con error positivo grande. ....                       | 72 |
| Figura 49 Simulación del control difuso con error negativo pequeño.....                       | 72 |
| Figura 50 Simulación del control difuso con error muy cercano a cero. ....                    | 73 |
| Figura 51 Instalación de librería numpy y scikit-fuzzy.....                                   | 74 |
| Figura 52 Instalación de librerías en la Raspberry.....                                       | 74 |
| Figura 53 Código de control difuso parte 1 en Visual Studio en lenguaje Python.....           | 75 |
| Figura 54 Código de control difuso parte 2 en Visual Studio en lenguaje Python.....           | 75 |
| Figura 55 Código de control difuso parte 3 en Visual Studio en lenguaje Python.....           | 76 |
| Figura 56 Código de control difuso parte 4 en Visual Studio en lenguaje Python.....           | 76 |
| Figura 57 Código de control difuso parte 4 en Visual Studio en lenguaje Python.....           | 76 |
| Figura 58 Simulación de control difuso en la terminal de la laptop, desde Visual Studio. .... | 77 |
| Figura 59 Ejecución desde la terminal de la Raspberry pi. ....                                | 77 |
| Figura 60 Simulación realizada en Raspberry pi. ....                                          | 78 |
| Figura 61 Diagrama del circuito electrónico conectado a la Raspberry pi. ....                 | 79 |
| Figura 62 Vista frontal de prototipo en 3D realizado con medidas reales. ....                 | 80 |
| Figura 63 Vista lateral de prototipo en 3D con medidas reales.....                            | 81 |
| Figura 64 Vista inferior de prototipo en 3D con medidas reales. ....                          | 81 |
| Figura 65 Vista superior de prototipo en 3D con medidas reales.....                           | 82 |
| Figura 66 Código de interfaz táctil de usuario parte 1 en Visual Studio con Python ....       | 83 |
| Figura 67 Código de interfaz táctil de usuario parte 2 en Visual Studio con Python ....       | 83 |
| Figura 68 Código de interfaz táctil de usuario parte 3 en Visual Studio con Python ....       | 84 |
| Figura 69 Código de interfaz táctil de usuario parte 4 en Visual Studio con Python ....       | 84 |
| Figura 70 Código de interfaz táctil de usuario parte 5 en Visual Studio con Python ....       | 85 |
| Figura 71 Código de interfaz táctil de usuario parte 6 en Visual Studio con Python ....       | 85 |
| Figura 72 Código de interfaz táctil de usuario parte 7 en Visual Studio con Python ....       | 86 |
| Figura 73 Código de interfaz táctil de usuario parte 8 en Visual Studio con Python ....       | 86 |
| Figura 74 Código de interfaz táctil de usuario parte 9 en Visual Studio con Python ....       | 87 |
| Figura 75 Código de interfaz táctil de usuario parte 10 en Visual Studio con Python ....      | 87 |
| Figura 76 Interfaz de usuario.....                                                            | 88 |
| Figura 77 Código de interfaz web de usuario parte 1 en Visual Studio con HTML.....            | 89 |
| Figura 78 Código de interfaz web de usuario parte 2 en Visual Studio con HTML.....            | 89 |
| Figura 79 Código de interfaz web de usuario parte 3 en Visual Studio con HTML.....            | 90 |
| Figura 80 Código de interfaz web de usuario parte 3 en Visual Studio con HTML.....            | 90 |
| Figura 81 Aplicación web. ....                                                                | 91 |
| Figura 82 Diagrama esquemático de Arduino UNO y caudalímetros. ....                           | 92 |
| Figura 83 Superficie alta para medir caudal.....                                              | 92 |
| Figura 84 Conexiones físicas de Arduino y caudalímetros.....                                  | 93 |
| Figura 85 Parte 1 de código en IDE de Arduino UNO.....                                        | 93 |
| Figura 86 Parte 2 de código en IDE de Arduino UNO.....                                        | 94 |

|                                                                                  |     |
|----------------------------------------------------------------------------------|-----|
| Figura 87 Parte 3 de código en IDE de Arduino UNO.....                           | 94  |
| Figura 88 Valores obtenidos por los caudalímetros. ....                          | 95  |
| Figura 89 Diagrama esquemático de caudalímetros y Rasberry Pi. ....              | 95  |
| Figura 90 Conexiones de Rasberry Pi y caudalímetros. ....                        | 96  |
| Figura 91 Instalación de librerías gpiozero,RPi GPIO. ....                       | 97  |
| Figura 92 Parte 1 de código en Visual Studio en lenguaje Python.....             | 97  |
| Figura 93 Parte 2 de código en Visual Studio en lenguaje Python.....             | 98  |
| Figura 94 Parte 3 de código en Visual Studio en lenguaje Python.....             | 98  |
| Figura 95 Parte 4 de código en Visual Studio en lenguaje Python.....             | 98  |
| Figura 96 Parte 5 de código en Visual Studio en lenguaje Python.....             | 99  |
| Figura 97 Comando de cambio de directorio. ....                                  | 99  |
| Figura 98 Ejecución del código de monitorización de flujo en la Rasperry pi..... | 100 |
| Figura 99 Diagrama Esquemático de Arduino UNO y motores nema 17. ....            | 101 |
| Figura 100 Conexiones físicos de Arduino UNO y motores nema 17. ....             | 102 |
| Figura 101 Agregamos la librería descargada.....                                 | 103 |
| Figura 102 Abrimos nuestra carpeta de archivos y buscamos el .ZIP.....           | 104 |
| Figura 103 Verificamos que se instale correctamente. ....                        | 104 |
| Figura 104 Parte 1 de código en IDE de Arduino UNO.....                          | 105 |
| Figura 105 Parte 2 de código en IDE de Arduino UNO.....                          | 105 |
| Figura 106 conexiones físicas de drivers y motores nema 17. ....                 | 106 |
| Figura 107 Diagrama esquemático de Rasperry pi, drivers, motores nema 17. ....   | 106 |
| Figura 108 Conexiones de Rasperry pi y drivers. ....                             | 107 |
| Figura 109 Conexión de drivers a motores nema 17. ....                           | 108 |
| Figura 110 Instalación de librerías.....                                         | 108 |
| Figura 111 Parte 1 de código en Visual Studio. ....                              | 109 |
| Figura 112 Parte 2 de código en Visual Studio. ....                              | 109 |
| Figura 113 Parte 3 de código en Visual Studio. ....                              | 110 |
| Figura 114 Parte 4 de código en Visual Studio. ....                              | 110 |
| Figura 115 Cambio de directorio a la carpeta donde está el archivo .py.....      | 111 |
| Figura 116 Ejecución del código.....                                             | 111 |
| Figura 117 Diagrama esquemático de Arduino UNO y sensores DS18B20. ....          | 112 |
| Figura 118 conexiones de Arduino UNO a sensores DS18B20. ....                    | 112 |
| Figura 119 Instalación de librería OneWire. ....                                 | 113 |
| Figura 120 Instalación de librería DallasTemperature. ....                       | 113 |
| Figura 121 Código 1 para identificar los sensores.....                           | 114 |
| Figura 122 Direcciones de los 3 sensores. ....                                   | 115 |
| Figura 123 Identificadores para poder medir la temperatura.....                  | 115 |
| Figura 124 Dirección de sensores y temperaturas registradas.....                 | 116 |
| Figura 125 Diagrama esquemático de sensores y Rasberry Pi. ....                  | 117 |
| Figura 126 Conexiones de sensores y Rasberry Pi. ....                            | 117 |
| Figura 127 Instalación de librerías.....                                         | 118 |
| Figura 128 Comando a ejecutar. ....                                              | 118 |
| Figura 129 Comando en Rasberry Pi.....                                           | 118 |
| Figura 130 Cambios en el archivo.....                                            | 119 |

|                                                                                              |     |
|----------------------------------------------------------------------------------------------|-----|
| Figura 131 Reinicio de la Raspberry Pi .....                                                 | 119 |
| Figura 132 Búsqueda de dispositivos.....                                                     | 120 |
| Figura 133 Parte 1 de código para agregar las direcciones de los sensores.. .....            | 120 |
| Figura 134 Parte 2 de código con las direcciones de los sensores. ....                       | 121 |
| Figura 135 Parte 3 de código de los sensores en Visual Studio en lenguaje Python. ....       | 121 |
| Figura 136 Cambio de directorio a donde se encuentra la carpeta. ....                        | 122 |
| Figura 137 Mediciones de los sensores usando una fuente de calor. ....                       | 122 |
| Figura 138 Variación de fuente de calor en los sensores y sus respectivas temperaturas. .... | 123 |
| Figura 139 MDF 9mm.....                                                                      | 124 |
| Figura 140 Corte de sección en MDF. ....                                                     | 125 |
| Figura 141 Corte de las piezas. ....                                                         | 125 |
| Figura 142 Clavos de una pulgada. ....                                                       | 126 |
| Figura 143 Pegamento blanco 850. ....                                                        | 126 |
| Figura 144 Estructura de MDF.....                                                            | 127 |
| Figura 145 Lija de grano 100.....                                                            | 128 |
| Figura 146 Pieza sutilmente lijada de la parte interna. ....                                 | 128 |
| Figura 147 Pieza sutilmente lijada de la parte externa. ....                                 | 129 |
| Figura 148 Comprobación de las medidas de a las medidas reales de cada elemento. ....        | 129 |
| Figura 149 Comprobación de medidas en estructura de MDF.....                                 | 130 |
| Figura 150 válvula de paso con conexiones de tubo y codo. ....                               | 131 |
| Figura 151 Conexión de caudín con tubos y conexiones necesarias. ....                        | 131 |
| Figura 152 Conexión de ambos lados del caudalímetro. ....                                    | 132 |
| Figura 153 Orificio en el tapón donde ira el sensor DS18B20.....                             | 132 |
| Figura 154 Ingreso del sensor DS18B20 en el orifico .....                                    | 133 |
| Figura 155 Aislamiento del orificio del sensor DS18B20.....                                  | 134 |
| Figura 156 Sensor DS18B20 aislado correctamente.....                                         | 134 |
| Figura 157 caudalímetro adherido correctamente. ....                                         | 135 |
| Figura 158 Unión de la válvula a los tubos y codos con cemento. ....                         | 136 |
| Figura 159 Sensor de temperatura en su pieza de tubería.....                                 | 136 |
| Figura 160 Sistema hidráulico adherido correctamente. ....                                   | 137 |
| Figura 161 Bigote de CCPVC. ....                                                             | 138 |
| Figura 162 Se adhiere el lado derecho del sistema hidráulico. ....                           | 138 |
| Figura 163 Se adhiere el lado izquierdo del sistema hidráulico. ....                         | 139 |
| Figura 164 Se adhiere la te de CCPVC que conecta al bigote con el sensor. ....               | 139 |
| Figura 165 Sistema hidráulico y electrónico adherido parcialmente. ....                      | 140 |
| Figura 166 Sistema adherido a la estructura de MDF.....                                      | 141 |
| Figura 167 Tarraja M5 de 0.8mm .....                                                         | 142 |
| Figura 168 Tarraja y motor nema 17.....                                                      | 142 |
| Figura 169 Tarraja colocada para girar. ....                                                 | 143 |
| Figura 170 Acoplamiento de válvula y motor nema.....                                         | 143 |
| Figura 171 Sistema hidráulico y electrónico finalizado. ....                                 | 144 |
| Figura 172 Error negativo grande.....                                                        | 146 |
| Figura 173 Error negativo pequeño. ....                                                      | 146 |
| Figura 174 Error cero.....                                                                   | 147 |

|                                                                                       |     |
|---------------------------------------------------------------------------------------|-----|
| Figura 175 Error positivo pequeño. ....                                               | 147 |
| Figura 176 Error positivo grande. ....                                                | 148 |
| Figura 177 Cerrado. ....                                                              | 149 |
| Figura 178 Poco abierto. ....                                                         | 149 |
| Figura 179 Medio abierto. ....                                                        | 150 |
| Figura 180 Casi abierto. ....                                                         | 150 |
| Figura 181 Abierto. ....                                                              | 151 |
| Figura 182 Variables de control difuso. ....                                          | 152 |
| Figura 183 Error negativo grande. ....                                                | 153 |
| Figura 184 Error negativo pequeño. ....                                               | 153 |
| Figura 185 Error cercano a cero. ....                                                 | 154 |
| Figura 186 Error positivo pequeño. ....                                               | 154 |
| Figura 187 Error positivo grande. ....                                                | 155 |
| Figura 188 Vista frontal del modelo de la estructura de MDF en 3D. ....               | 156 |
| Figura 189 Vista superior del modelo de la estructura de MDF en 3D. ....              | 157 |
| Figura 190 Vista lateral del modelo de la estructura de MDF en 3D. ....               | 157 |
| Figura 191 Vista posterior del modelo de la estructura de MDF en 3D. ....             | 158 |
| Figura 192 Elementos del sistema hidráulico – electrónico en el modelo 3D. ....       | 159 |
| Figura 193 Vista frontal del prototipo modelado en 3D con medidas. ....               | 160 |
| Figura 194 Vista lateral del prototipo modelado en 3D con medidas. ....               | 160 |
| Figura 195 Estructura MDF. ....                                                       | 161 |
| Figura 196 Conexiones de elementos electrónicos. ....                                 | 162 |
| Figura 197 Conexiones en Raspberry Pi. ....                                           | 163 |
| Figura 198 Inicialización del sistema con duración de 30 seg. ....                    | 164 |
| Figura 199 Válvulas cerradas al inicio de la prueba. ....                             | 164 |
| Figura 200 Sistema iniciado con temperaturas deseada y sensada iguales. ....          | 165 |
| Figura 201 Válvulas abiertas a la mitad. ....                                         | 165 |
| Figura 202 Sistema en funcionamiento con temperaturas deseada y sensada iguales. .... | 166 |
| Figura 203 Válvulas abiertas a la mitad. ....                                         | 166 |
| Figura 204 Ligero cambio en la temperatura sensada. ....                              | 167 |
| Figura 205 Válvulas abiertas a la mitad debido a que el error es cercano a cero. .... | 167 |
| Figura 206 Reducción de la temperatura deseada a 22°C. ....                           | 168 |
| Figura 207 Cambio en la apertura de las válvulas. ....                                | 168 |
| Figura 208 Temporizador en cero y temperatura deseada igual a la sensada. ....        | 169 |
| Figura 209 Válvulas cerradas por completo. ....                                       | 169 |
| Figura 210 Introducir tiempo. ....                                                    | 170 |
| Figura 211 Introducir temperatura. ....                                               | 171 |
| Figura 212 Aplicación web en curso. ....                                              | 172 |
| Figura 213 Aumento en la temperatura sensada. ....                                    | 172 |
| Figura 214 Prueba de la aplicación web en laptop. ....                                | 173 |

## **Capítulo I. Generalidades del proyecto.**

### **1.1 Introducción.**

La domótica ha revolucionado la forma en que interactuamos con nuestro entorno en el hogar, llevando la automatización y la conectividad a diferentes lugares. Desde la seguridad hasta la comodidad. Las duchas inteligentes emergen como una innovación destacada, prometiendo una experiencia de baño personalizada y conveniente.

Una vivienda se considera verdaderamente domótica cuando integra dispositivos automatizados que se comunican entre sí, aprovechando la interconexión de los sistemas de información y comunicación (TIC). De esta manera las duchas inteligentes junto con la tecnología pueden transformar rutinas diarias.

La ducha inteligente va más allá de simplemente regular la temperatura del agua; se controla mediante asistentes virtuales, ajusta la temperatura automáticamente según las preferencias del usuario. Con cada función diseñada para mejorar la experiencia de baño, las duchas inteligentes representan la convergencia perfecta entre confort y tecnología.

El presente proyecto se suma a esta evolución, proponiendo un sistema de control difuso que regula la temperatura del agua en tiempo real. Además, busca desarrollar una aplicación intuitiva que simplifique la interacción del usuario con la ducha inteligente, ofreciendo herramientas atractivas que mejoren aún más la experiencia de baño del usuario. En un mundo donde la tecnología redefine constantemente nuestras vidas, las duchas inteligentes son un claro ejemplo de cómo la innovación puede transformar hasta los aspectos más cotidianos de nuestro hogar.

## **1.2 Datos de la empresa.**

### **Tecnológico Nacional de México / Instituto Tecnológico de Apizaco.**

El Instituto Tecnológico de Apizaco ofrece en su formación educativa 9 Ingenierías, 3 Maestrías y un Doctorado, cuyas matrículas abarcan de manera integral una formación general básica que incluye el fortalecimiento científico y metodológico; todas las carreras cuentan con especialidades diseñadas para atender las necesidades de los sectores productivo y social. Los planes de estudio están basados en una estructura genérica, una especialidad y una residencia profesional.

#### **MISIÓN:**

Formar profesionistas que sean líderes visionarios, emprendedores, con capacidad de generar soluciones disruptivas, innovando en la ciencia y la tecnología, comprometidos con el desarrollo económico, responsabilidad social teniendo una visión sostenible y sustentable.

#### **VISIÓN:**

Ser una Institución de educación superior tecnológica de vanguardia, formadora de ciudadanos del mundo, con conocimientos intercultural, respetando la diversidad social.

#### **VALORES:**

Honestidad: Se demuestra día a día a través de la transparencia con la ejecución de las actividades laborales cotidianas.

Responsabilidad: Es la actitud que caracteriza al personal en el cumplimiento de su tarea con eficacia y eficiencia en beneficio de la comunidad.

Lealtad: Es la actitud que caracteriza al personal del Instituto Tecnológico de Apizaco, aplica sus conocimientos y habilidades en el logro de las metas institucionales, sin olvidar en ningún momento el lema:

Organigrama. En la figura 1 y 2 se muestra la estructura organizativa y el organigrama de la empresa respectivamente.

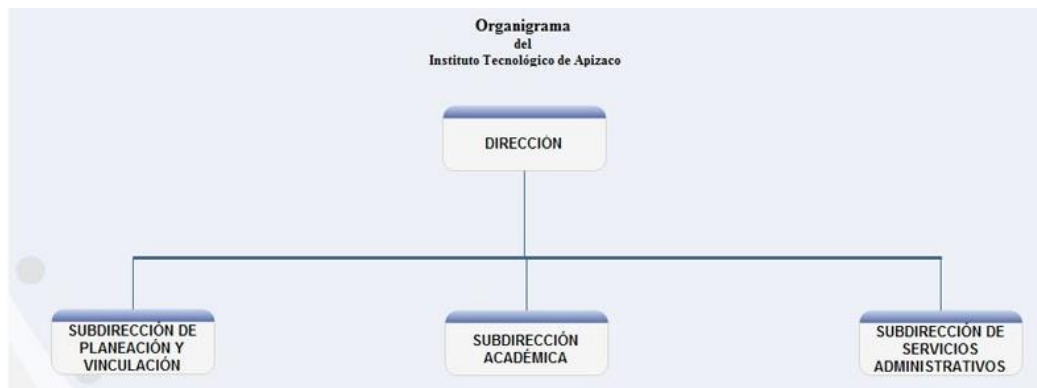


Figura 1 Estructura Organizativa de Instituto Tecnológico de Apizaco.

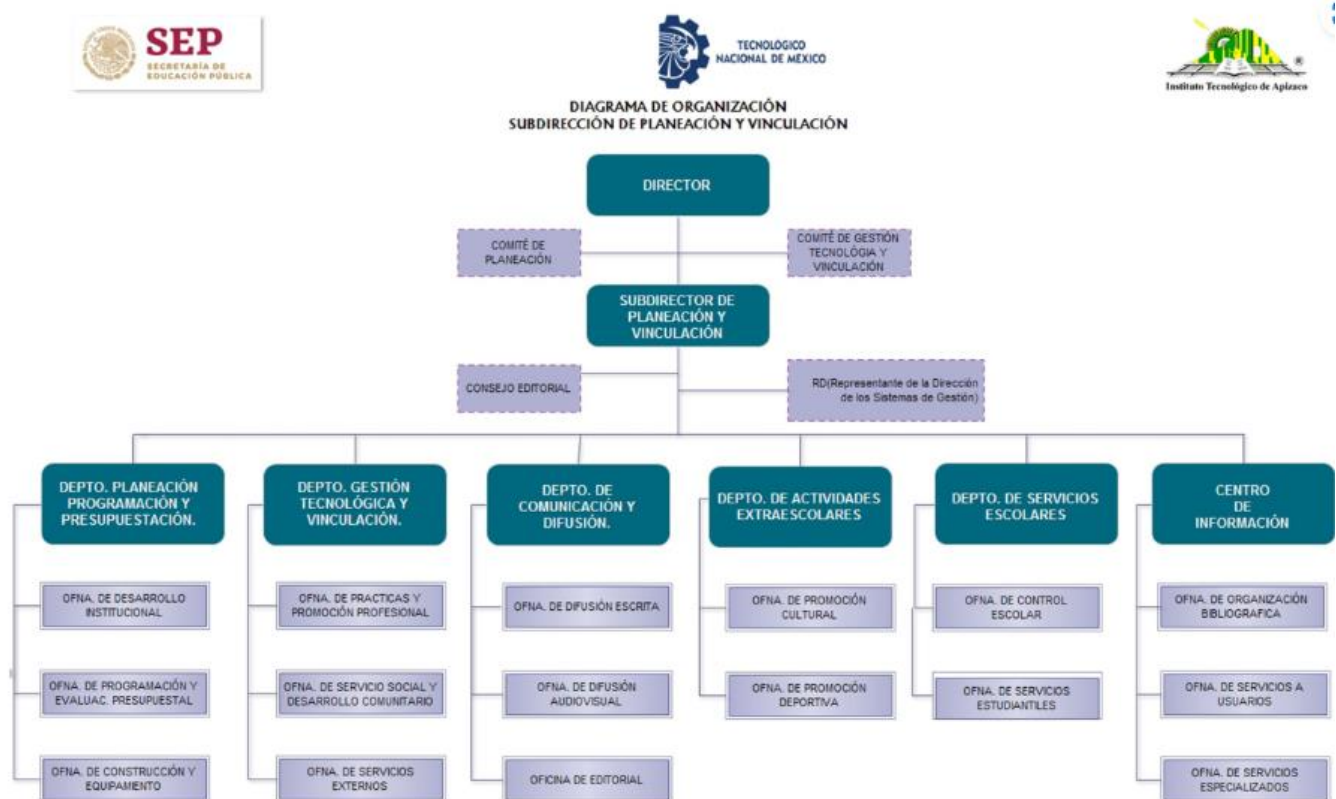


Figura 2 Organigrama de Instituto Tecnológico de Apizaco



## Ubicación Geográfica.

Dentro de las figuras 3 y 4 se da a conocer la ubicación geográfica de la institución, en sus dos campus.



*Figura 3 Ubicación de Instituto Tecnológico de Apizaco.*



*Figura 4 Ubicación de Instituto Tecnológico de Apizaco – Posgrado e Investigación.*

## UBICACIÓN

**Dirección:** Av. Instituto Tecnológico No. 418, San Andrés Ahuashuatepec, Municipio de Tzompantepec, Tlaxcala, Méx. C.P. 90491.

**RFC:** TNM140723GFA

**Ciudad:** Apizaco, Tlaxcala.

**teléfono:** 241 417 2010

**Correo Electrónico:** [dirección@apizaco.tecnm.mx](mailto:dirección@apizaco.tecnm.mx)

**Sitio Web:** <https://www.apizaco.tecnm.mx/>

### 1.2.1 Área del trabajo del estudiante.

El desarrollo del proyecto tuvo lugar en el Departamento de Ingeniería Eléctrica y Electrónica. El espacio asignado para el desarrollo del proyecto fue en el aula alfa 17 ubicada en el TecNM Campus Apizaco-Posgrado e investigación, la cual se muestra en las figuras 5 y 6.



*Figura 5 Aula alfa 17 Posgrado e investigación.*



*Figura 6 Área donde se desarrollaron las pruebas preliminares.*

### **1.3 Planteamiento del problema.**

La domótica ha experimentado una expansión notable, redefiniendo nuestra relación con el entorno del hogar mediante la introducción de sistemas automatizados y conectados. Dentro de esta evolución, las duchas inteligentes han surgido como una innovación destacada, prometiendo una experiencia de baño personalizada y cómoda. Sin embargo, para aprovechar al máximo el potencial de estas duchas inteligentes, es crucial desarrollar sistemas de control eficientes que regulen la temperatura del agua de manera óptima.

Por otra parte, muchas personas de edad avanzada o quienes presentan enfermedades de motricidad suelen requerir asistencia al momento de bañarse. Algunas enfermedades que afectan la motricidad y coordinación de distintas partes del cuerpo son, la distonía y distonía cervical, enfermedad de Huntington, mioclonía, Parkinson y parkinsonismo. Los trastornos que generan parkinsonismo son, la enfermedad de Alzheimer, la atrofia multisistémica, el deterioro cortico basal ganglionar, la demencia frontotemporal, la parálisis supranuclear progresiva, encefalitis vírica, tumores cerebrales y accidentes cerebrovasculares, traumatismo craneal y la enfermedad de Wilson. Todas estas enfermedades y trastornos pueden afectar al cuerpo completo o a zonas específicas como el tren superior o inferior, generándoles dificultades para manejar los dispositivos de aseo convencionales

Actualmente, las duchas inteligentes ofrecen funciones avanzadas, como la regulación automática de la temperatura del agua y la integración con asistentes virtuales para una mayor comodidad del usuario. Sin embargo, la implementación de un sistema de control difuso podría mejorar aún más esta experiencia al proporcionar un ajuste más preciso y adaptable en tiempo real. Además, la creación de una aplicación intuitiva para interactuar con la ducha inteligente podría simplificar la experiencia del usuario y brindar herramientas atractivas para personalizar aún más el baño.

Por lo tanto, el problema principal que aborda este proyecto es cómo diseñar e implementar un sistema de control difuso para regular la temperatura del agua en tiempo real en duchas inteligentes, junto con el desarrollo de una aplicación intuitiva para mejorar la experiencia de baño del usuario, con el fin de que pueda ser utilizado por cualquier tipo de personas, con

reducción de movilidad o no. Esta investigación busca explorar cómo la tecnología puede optimizar aún más los aspectos cotidianos del hogar, aprovechando la innovación para ofrecer un mayor confort y comodidad a los usuarios.

## **1.4 Objetivos.**

### **1.4.1 Objetivo general.**

Diseñar e implementar un sistema que sea capaz de monitorear y ajustar la temperatura de agua presente en sistemas hidráulicos domésticos, por medio de un panel de control y dispositivos Smart.

### **1.4.2 Objetivos específicos.**

- Realizar una investigación acerca de lo existente en la actualidad, relacionado con IoT y control de dispositivos.
- Realizar y comprobar el funcionamiento del programa, en simulación por medio del software Matlab y en emulación por medio de la Raspberry, programándola para el monitoreo, regulación y visualización de datos obtenidos a través de los sensores mediante del lenguaje de programación Python
- Desarrollar una interfaz de usuario en el teléfono que permita realizar ajustes en la ducha, como los del microprocesador en el panel.
- Diseñar las carcasas para el sistema hidráulico con funcionamiento manual y eléctrico, y para la unidad de procesamiento principal con aislamiento de humedad.
- Realizar pruebas al sistema para garantizar su funcionalidad en diferentes escenarios y condiciones.
- Documentar el proyecto de manera detallada, incluyendo el diseño, implementación, resultados y conclusiones.

## **1.5 Justificación.**

La implementación de sistemas domóticos en una vivienda representa una evolución significativa en la manera en que interactuamos con nuestro entorno residencial. Estos sistemas inteligentes se diseñan con el propósito de adaptarse de forma dinámica a las necesidades y preferencias de los usuarios, mejorando así la habitabilidad y el confort de los espacios habitacionales.

Uno de los beneficios más destacados de la domótica es su capacidad para optimizar el consumo energético y, por ende, generar un ahorro significativo a largo plazo. Esta eficiencia energética se traduce en una reducción de los costos asociados al mantenimiento de la vivienda, al tiempo que contribuye a la preservación del medio ambiente al minimizar el desperdicio de recursos.

En el contexto específico del baño, donde tradicionalmente se registra un alto consumo energético, la integración de sistemas domóticos cobra aún más relevancia. La capacidad de controlar la temperatura del agua de manera precisa, así como la implementación de sensores de movimiento y cámaras de seguridad, no solo mejoran la experiencia del usuario, sino que también garantizan un uso más responsable de los recursos, evitando posibles accidentes y contribuyendo al ahorro de agua y energía.

El prototipo será un dispositivo electrónico que las personas con discapacidad física podrán manejar sin necesidad de utilizar sus extremidades, generándoles confianza, seguridad e independencia al momento de bañarse; de esta manera se genera un impacto social, ya que la invención ofrece una gran facilidad de uso, pues contará con la ventaja de poder seleccionar y controlar la temperatura y presión del agua, todo desde un dispositivo móvil y desde el panel de control. De tal forma que pueda ser implementado en hospitales o en hogares con algún paciente de edad avanzada o con problemas de movilidad.

En entornos fuera del hogar, como la industria hotelera, la aplicación de tecnologías domóticas se convierte en un elemento clave para garantizar la satisfacción de los huéspedes y optimizar la gestión de los recursos. Desde el control automatizado de la iluminación hasta la implementación de sistemas de ducha inteligente, estas soluciones tecnológicas no solo mejoran la experiencia del cliente, sino que también permiten reducir los costos operativos y minimizar el impacto ambiental.

En última instancia, el uso racional de los recursos naturales se vuelve fundamental en un contexto donde el consumo excesivo de agua es una preocupación creciente. Estadísticas de organismos como la Organización Mundial de la Salud (OMS) y la Comisión Nacional del Agua (Conagua) resaltan la importancia de adoptar medidas para reducir el consumo de agua potable, especialmente en regiones como Tlaxcala, donde los índices de consumo son significativamente altos. En este sentido, la implementación de sistemas domóticos, incluidos los de ducha inteligente, no solo se presenta como una opción tecnológica avanzada, sino como una medida práctica y necesaria para promover la sostenibilidad y el uso responsable de los recursos hídricos.

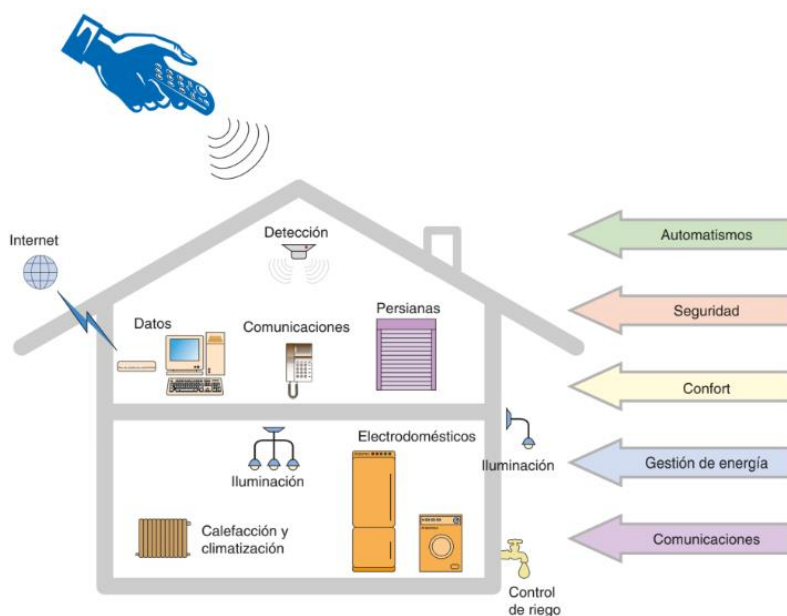


## Capítulo II. Marco teórico.

### 2.1 Definición de domótica.

La domótica, como técnica revolucionaria, despliega un abanico de posibilidades al permitir la automatización integral de los sistemas eléctricos tanto en hogares como en edificaciones de índole diversa. Aunque su aplicación inicialmente se enfocaba en entornos más complejos que las simples viviendas, su alcance se ha expandido para abarcar una gama amplia de instalaciones. En el ámbito doméstico, la implementación de sistemas que automatizan circuitos específicos ha sido una práctica común, con dispositivos autónomos que ejecutan diversas funciones como encender dispositivos programados, activar luces al detectar presencia, regular la intensidad lumínica y detectar cambios en magnitudes físicas, entre otras acciones.

No obstante, es crucial destacar que estos sistemas, aunque eficientes, operan de manera aislada, como islas independientes en el vasto mar de la automatización. La evolución natural de este campo tecnológico ha llevado a reconocer la necesidad imperiosa de integrar estas automatizaciones con otros servicios e instalaciones del hogar, dando origen al concepto de "domótica". En este sentido, los hogares dotados de una infraestructura domótica son comúnmente denominados "Smart Homes" o "Hogares Inteligentes", figura 7.



*Figura 7 Esquema de una vivienda domótica.*

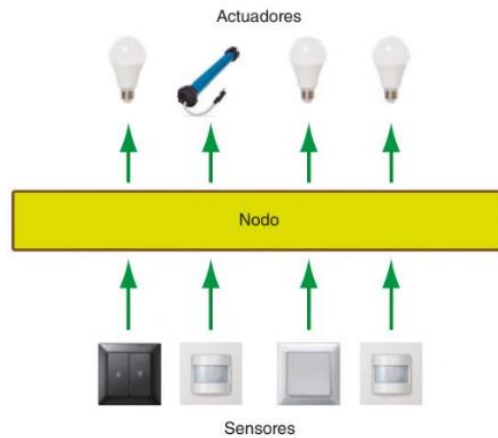
La integración exitosa de otros servicios del hogar en una red domótica implica una incorporación plena en el entramado eléctrico de la vivienda, facilitando así la convergencia de diversos sistemas, como redes de datos, seguridad y multimedia. Aunque estos servicios podrían funcionar de manera autónoma, su integración con sistemas domóticos potencia su gestión y eficacia operativa, ofreciendo una experiencia sinérgica que optimiza el funcionamiento general del hogar y mejora la calidad de vida de sus habitantes.

Los servicios tienen la capacidad de operar de manera independiente, sin embargo, cuando se integran con sistemas domóticos, se potencia su eficiencia y rendimiento. La sinergia entre estos servicios y las instalaciones domóticas no solo permite una gestión óptima, sino que también abre la puerta a nuevas posibilidades de automatización y control, mejorando así la experiencia del usuario y aumentando la eficacia en el uso de recursos.

La convergencia de estos elementos ofrece un entorno más inteligente y adaptable, donde los servicios se ajustan de manera dinámica a las necesidades cambiantes, brindando una mayor comodidad y seguridad en el hogar o en cualquier otro entorno donde se aplique esta tecnología [1].

Domótica es un término que se emplea para denominar la parte de la tecnología (electrónica e informática), que integra el control y supervisión (automatización) de los elementos existentes en un edificio de oficinas o en cualquier hogar. Un sistema domótico consta de una red de comunicaciones que permite la interconexión de una serie de equipos para obtener información sobre el entorno doméstico y realizar acciones sobre dicho entorno.

Los elementos de campo (detectores, sensores, captadores, actuadores, etc.), transmitirán las señales a una unidad central inteligente para tratar la información recibida, ejemplificados en la figura 8. En función de dicha información y de una determinada programación, la unidad central actúa sobre determinados circuitos de potencia relacionados con las señales recogidas por estos elementos [2].



*Figura 8 Actuadores y sensores.*

La domótica, al centrarse en la interconexión de sus distintos componentes, a menudo encuentra aplicaciones dentro del ámbito del Internet de las Cosas (IoT). Este concepto implica la capacidad de conectar elementos físicos cotidianos a la red de internet, abarcando desde objetos domésticos convencionales, como lámparas, hasta recursos médicos, como dispositivos de monitorización de la salud, así como prendas y accesorios personales inteligentes. Incluso los sistemas utilizados en el desarrollo de ciudades inteligentes encuentran su lugar en esta interconexión tecnológica. Esta convergencia entre la domótica y el IoT no solo amplía las posibilidades de automatización y control en el entorno doméstico y urbano, sino que también promueve la eficiencia, la comodidad y la seguridad para los usuarios. Además, posibilita la creación de ecosistemas más inteligentes y adaptativos que pueden mejorar significativamente la calidad de vida de las personas [3].

## 2.2 Dispositivos del IoT.

Estos dispositivos se encuentran dentro de esos objetos físicos (figura 9) y suelen pertenecer a una de estas dos categorías: interruptores (es decir, envían las instrucciones a un objeto) o sensores (recopilan los datos y los envían a otro lugar). Los dispositivos IoT hacen referencia a los sistemas de dispositivos físicos que reciben y transfieren datos a través de las redes inalámbricas,

con poca intervención humana. Los sistemas de dispositivos IoT son posibles gracias a la integración de dispositivos informáticos en todo tipo de objetos [4].



*Figura 9 Dispositivos IoT.*

### **2.3 Conexión entre dispositivos IoT.**

Los dispositivos pertenecientes al Internet de las Cosas (IoT) son objetos que se conectan de manera inteligente entre sí mediante una combinación de sensores inalámbricos, software especializado y otras tecnologías avanzadas, como se muestra en la figura 10. Estos dispositivos emplean una variedad de componentes, como sensores accesibles y de bajo consumo, protocolos de red para internet (como wifi o redes móviles), plataformas de computación en la nube (Cloud Computing), análisis de datos y técnicas de aprendizaje automático (machine learning), así como inteligencia artificial (IA). Un solo dispositivo puede estar equipado con múltiples sensores, como un teléfono inteligente que cuenta con GPS, cámara, acelerómetro, entre otros. Estos sensores tienen la función principal de recopilar datos del entorno con el fin de cumplir un propósito específico, ya sea proporcionar información sobre la ubicación, capturar imágenes o medir variables como la temperatura o la presión. Esta capacidad de recolección de datos es fundamental para el funcionamiento eficiente y la utilidad práctica de los dispositivos IoT en una amplia gama de aplicaciones, desde la domótica hasta la monitorización de la salud y la gestión urbana inteligente. La integración de estas tecnologías permite que los dispositivos IoT sean cada vez más versátiles y adaptables, mejorando así nuestra capacidad para interactuar con el mundo que nos rodea de manera más eficiente y eficaz [5].



*Figura 10 Conexión entre dispositivos IoT.*

## **2.4 Aplicación IoT**

Los dispositivos IoT que se utilizan en domótica, es siempre necesaria una aplicación de IoT, la cual es un conjunto de servicios y software que integra los datos recibidos de varios dispositivos de IoT. Utiliza tecnología de machine learning o inteligencia artificial (IA) para analizar estos datos y tomar decisiones informadas. Estas decisiones se comunican al dispositivo de IoT y este responde de forma inteligente a las entradas generadas por los sensores presentes en los dispositivos IoT [6].

## **2.5 Interfaz de usuario.**

Las aplicaciones del Internet de las Cosas (IoT) necesitan una interfaz de usuario (UI) para facilitar la interacción entre las personas y los dispositivos o software. Esta interfaz actúa como el canal mediante el cual los usuarios controlan y acceden a las funcionalidades de una aplicación o dispositivo, brindando controles gráficos que optimizan la experiencia del usuario y permiten la ejecución de diversas tareas. Es esencial que los equipos de desarrollo de la interfaz de usuario prioricen la apariencia del software, ya que esto contribuye a que los usuarios se sientan familiarizados y cómodos con el programa. La presentación de la información debe ser clara y precisa para evitar confusiones o errores durante la interacción, por lo que todos los elementos de la aplicación deben mantener coherencia en su diseño y propósito. Cuando se logra este equilibrio, los usuarios pueden adoptar patrones de uso de manera intuitiva, sin la necesidad de

aprender procesos complicados que podrían dificultar su experiencia con la tecnología. En definitiva, una UI bien diseñada no solo mejora la usabilidad de las aplicaciones IoT, sino que también promueve una conexión más fluida y eficiente entre las personas y la tecnología que utilizan en su vida cotidiana [7].

## **2.6 Lógica difusa.**

La lógica difusa se puede definir como un sistema interpretativo en el que los objetos o elementos se relacionan con conjuntos no claramente definidos, asignándoles un grado de pertenencia relativo o graduado en lugar de una pertenencia estricta, como se observa en la lógica convencional. En un sentido más amplio, se establece una especie de transición gradual entre fronteras o entre conjuntos. Esto nos permite incorporar declaraciones del lenguaje común, las cuales suelen ser ambiguas, para interpretar el estado de las variables de un proceso determinado, asignándoles en cada momento un grado de pertenencia a estos conjuntos difusos. Esto se lleva a cabo con el fin de diseñar un controlador que, de manera intuitiva (no necesariamente numéricamente exacta), interprete el estado de ciertas variables y, basándose en ello, deduzca de manera lógica una acción posible que permita llevar la variable al estado deseado [8].

El control Fuzzy comprende diversas fases que incluyen:

- **Etapas de Fusificación:** En este paso, se asigna a los datos de entrada un valor dentro de las expresiones posibles, donde el grado de pertenencia de una variable es mayor en una expresión que en las demás. Una vez que se ha establecido el estado de las variables en forma lingüística, se pueden establecer relaciones lógicas entre ellas mediante reglas IF-THEN.
- **Funciones de Pertenencia:** Estas son curvas que definen el grado de pertenencia de las variables a los conjuntos difusos, expresado entre uno y cero. Se definen según la experiencia y análisis del sistema a controlar, y pueden tomar formas como: trapezoidal, triangular, gaussiana, entre otras.

- **Variables de Entrada:** En el diagrama de bloques del control Fuzzy, la primera variable de entrada suele ser el error, que representa la diferencia entre el valor de referencia y los datos del sistema o proceso.
- **Variables de Salida:** Estas variables describen tanto la salida de potencia del sistema como el control de temperatura, por ejemplo.
- **Reglas de Control:** Se definen utilizando una notación como "Muy negativo", "Negativo", "Cero", "Positivo" y "Muy positivo".
- **Funciones de Pertenencia:** Cada función de pertenencia debe tener un nombre único que coincida con el valor asignado en la tabla de reglas de control.
- **Desfusicación:** En esta etapa, se representa numéricamente el resultado de los grados de pertenencia de las variables de entrada. Se evalúan las reglas de inferencia y se obtiene la respuesta del controlador en términos de cada conjunto de salida. El método de defusicación utilizado en el diseño del controlador Fuzzy es el de centro de masa, también conocido como método de centro de gravedad. Este método es ampliamente utilizado ya que retorna el centro del área de la curva o del polígono generado en la etapa de inferencia [9].

## **2.7 Sensor de temperatura.**

Un sensor de temperatura es un dispositivo que típicamente convierte un valor de temperatura en una señal eléctrica que puede ser registrada [10].

También conocido como sonda de temperatura o termosensor, este dispositivo despliega una variedad de tipos, tales como:

- **Termopares:** Estos sensores miden la temperatura utilizando dos metales diferentes unidos en un extremo. Cuando la unión de los metales se calienta o enfría, se genera una tensión que es proporcional a la temperatura.
- **Termorresistencias:** En estos sensores, se emplean resistencias dado que la conductividad cambia según la temperatura. Esta variación permite determinar la temperatura del material.

Los sensores de temperatura varían en su apariencia o construcción según su tarea específica y lugar de instalación. En este sentido, encontramos diversos tipos:

- Sensores de inmersión: Estos realizan la medición directamente en el medio a ser medido, como agua, aceite, gas o aire.
- Sensores de cable: Utilizan un elemento de medición generalmente encerrado en una cubierta metálica al final de un cable.
- Sensores de aplicación: Realizan una medición indirecta de la temperatura en el exterior de una aplicación.
- Sensores de temperatura de tubería con clip: Ofrecen una medición indirecta de la temperatura de un medio, como el agua. Se montan en el exterior de una tubería, lo que ahorra costos y facilita la instalación.

### **2.7.1 Sensor de temperatura DS18B20**

El DS18B20 (mostrado en la figura 11) es un sensor de temperatura que es capaz de realizar la modificación analógica a digital dentro del encapsulado, lo que simplifica proceso para la conversión por medio de una interfaz digital conocida como 1-wire. Gracias a que cada DS18B20 cuenta con un número de serie de silicio único, es posible tener varios de estos sensores en el mismo bus de un solo hilo [11]. Esto posibilita la colocación de los sensores de temperatura en múltiples ubicaciones diferentes, permitiendo la lectura de la temperatura a través de una única conexión de cable (junto con tierra) desde un microprocesador central. Este sensor puede funcionar con sistemas que operan en un rango de voltaje de 3A y 5.5V.

El DS18B20 está disponible en diversas formas, siendo dos de las más comunes el encapsulado TO-92, utilizado típicamente en transistores, y el sensor tipo sumergible. Este último es fabricado por terceros, quienes colocan los sensores en encapsulado TO-92 dentro de una cubierta impermeable de acero inoxidable (ver anexo 1).



Las especificaciones del DS18B20 incluyen:

- Voltaje de alimentación/datos de 3A y 5,5 V.
- Precisión de  $\pm 0.5^{\circ}\text{C}$  en un rango de temperatura de  $-10^{\circ}\text{C}$  a  $+85^{\circ}\text{C}$ .
- Rango de temperatura utilizable de  $-55$  a  $125^{\circ}\text{C}$  ( $-67^{\circ}\text{F}$  a  $+257^{\circ}\text{F}$ ).
- Resolución de 9 a 12 bits.
- Utilización de una interfaz de un solo cable, lo que requiere solo un pin digital para la comunicación.
- ID especial de 64 bits.
- Posibilidad de que varios sensores compartan un pin.
- Sistema de alarma de límite de temperatura.
- Intervalo de tiempo menor a 750 ms.
- Interfaz de 3 hilos, incluyendo tipo A con cables rojo para VCC, negro para GND, y amarillo para datos.



*Figura 11 Sensor de temperatura DS18B20*

## **2.8 Caudalímetro**

Un caudalímetro es un dispositivo fundamental para la medición precisa del flujo de agua en sistemas de tuberías. Este sensor, situado en línea con la tubería, incorpora un molinete

especializado que registra la cantidad exacta de líquido que circula a través de él. Dotado de un sensor de efecto Hall magnético, emite impulsos eléctricos en cada giro del molinete. La tecnología de estos sensores se basa en el uso de ondas acústicas y campos electromagnéticos para calcular el flujo en una zona específica, aprovechando magnitudes físicas como la aceleración, la frecuencia, la presión y el volumen. Cada vez que una determinada cantidad de agua atraviesa la tubería, el sensor proporciona una lectura digital precisa.

El diseño del sensor de flujo de agua implica una válvula de plástico por la que fluye el agua, junto con un rotor y un sensor de efecto Hall que determina tanto la dirección como la velocidad del flujo. Cuando el agua pasa por la válvula, el rotor se pone en movimiento, lo que se traduce en cambios en la velocidad del motor. Estos cambios son interpretados como señales de pulso por el sensor de efecto Hall, permitiendo así la medición precisa de la tasa de flujo de agua.

El principio de funcionamiento subyacente de este dispositivo se basa en el efecto Hall. De acuerdo con este principio, la rotación del rotor induce una diferencia de voltaje en el conductor. Esta diferencia de voltaje inducida se produce perpendicularmente a la corriente eléctrica, lo que permite una detección precisa del flujo de agua. Este enfoque garantiza mediciones confiables y precisas, lo que resulta fundamental en una amplia gama de aplicaciones, desde la monitorización de sistemas de suministro de agua hasta el control de procesos industriales [12].

### **2.8.1 Caudalímetro YF-S201**

El dispositivo conocido como sensor de flujo o caudalímetro desempeña una función crucial en la determinación del flujo de agua en una tubería. Ubicado en línea con la corriente de agua, este sensor incorpora un mecanismo de molinete para cuantificar la cantidad de líquido que circula a través de él. Además, integra un sensor de efecto Hall magnético que genera un impulso eléctrico con cada rotación.

La operación de este caudalímetro, como el YF-S201 (figura 12), es esencialmente un proceso de detección y medición. El agua, al pasar por el sensor, acciona una turbina conectada a un imán. Este imán, al girar, activa el sensor de efecto Hall, desencadenando así un pulso eléctrico. Este impulso eléctrico puede ser fácilmente interpretado por dispositivos digitales como Arduino o PLC. Es relevante destacar que el sensor de efecto Hall está aislado del agua para garantizar su integridad y evitar daños.

Gracias a que el volumen de agua por cada pulso es constante y conocido, podemos calcular la cantidad de pulsos en una unidad de tiempo, como segundos o minutos. Al multiplicar este valor por el número total de pulsos, podemos obtener con precisión el caudal o flujo de agua.

El caudalímetro YF-S201, además de su funcionalidad, presenta una serie de características notables que lo hacen idóneo para diversas aplicaciones. Entre estas se incluyen su amplio rango de voltaje de operación, su capacidad para generar una onda cuadrada pulsante, y su impresionante precisión y durabilidad. Además, su diseño compacto y sus conexiones de tubería estandarizadas facilitan su instalación en diferentes entornos y aplicaciones (ver anexo 2).

Con todas estas cualidades, el caudalímetro YF-S201 emerge como una opción confiable y versátil para una amplia gama de aplicaciones que requieren mediciones precisas del flujo de agua [13].



*Figura 12 Caudalímetro YF-S201.*

## **2.9 Nema 17 Motor A Pasos 17HS4401**

El motor Nema 17 es un componente fundamental en el ámbito de la robótica y la automatización industrial, gracias a su tamaño estándar de 42 mm x 42 mm, que lo convierte en una opción compacta y versátil para integrar en una variedad de sistemas. Este motor está conformado por un rotor y un estator, donde el rotor contiene imanes permanentes y el estator aloja bobinas de alambre. Cuando se aplica corriente eléctrica a las bobinas, se crea un campo magnético que interactúa con los imanes del rotor, induciendo movimientos precisos en pasos discretos (ver anexo 3).

Características destacadas:

- Compacto y fácil de integrar: Su tamaño reducido facilita su incorporación en diseños y sistemas de dimensiones más reducidas.
- Alta precisión: Ofrece un nivel sobresaliente de precisión en el posicionamiento, ideal para aplicaciones que demandan movimientos exactos.
- Par de retención confiable: Presenta un buen par de retención, lo que significa que puede mantener una posición incluso sin corriente aplicada, siendo útil en aplicaciones que requieren bloqueo de posición.
- Control sencillo: Su manejo es fácil mediante controladores de motor paso a paso, permitiendo la regulación precisa de dirección y velocidad.
- Velocidad de respuesta rápida: Con una alta velocidad de respuesta, es capaz de acelerar y desacelerar de forma eficiente.
- Consumo energético bajo: A pesar de su potencia, su consumo energético es reducido, lo que resulta en una operación eficiente y costos operativos más bajos.
- Compatibilidad versátil: Se puede integrar con una amplia variedad de controladores y sistemas de control, incluyendo plataformas como Arduino, Raspberry Pi y otros microcontroladores.

El motor Nema 17, mostrado en la figura 17, es una elección popular en la industria debido a su combinación de características, lo que lo hace idóneo para una variedad de aplicaciones. Su facilidad de integración, precisión, eficiencia energética y capacidad de control lo convierten en una opción versátil y confiable para proyectos de robótica y automatización [14].



*Figura 13 Nema 17 Motor a pasos 17HS4401*

### **2.9.1 Driver TB6600**

El TB6600 de la figura 14 se presenta como una solución fácil de usar para controlar motores paso a paso de dos fases, ofreciendo una compatibilidad amplia con plataformas como Arduino y otros microcontroladores que puede generar señales de pulso digital de 5V (ver anexo 4).

Este controlador está equipado con 6 interruptores DIP, permitiendo una configuración versátil tanto de la corriente de salida, que abarca un rango desde 0.5A hasta 3.5A ajustable, como de los micropasos del motor, que pueden ser seleccionados entre 1, 2 / A, 2 / B, 4, 8, 16, y 32. Además, cuenta con conectores diseñados para facilitar la conexión del motor a pasos, la fuente de alimentación y los pines de control, lo que simplifica el proceso de instalación y configuración.

Entre sus especificaciones técnicas, el TB6600 ofrece un amplio rango de voltaje de alimentación de 9 a 42Vdc, con una corriente de entrada que varía desde 1A hasta un máximo de 5A. La señal de control oscila entre 3.3 a 24V, y la potencia máxima alcanza los 160W. Con una amplia gama de micro pasos ajustables, el TB6600 garantiza un movimiento suave y preciso del motor, mientras que su diseño compacto de 96x56x33 mm lo hace fácilmente integrable en diferentes sistemas y aplicaciones.

Además, este controlador está equipado con características de protección como protección contra sobrecalentamiento, sobrecorriente y cortocircuito, así como una entrada de protección anti-reversa. Su temperatura de funcionamiento oscila entre -10 y 45 °C, lo que garantiza un rendimiento estable en una variedad de condiciones ambientales.

Es importante tener en cuenta que, si bien el TB6600 ofrece una alta transmisión de corriente y puede proporcionar un alto par de torque al motor cuando está en movimiento, no es adecuado para aplicaciones que requieran altos niveles de par de retención estático. Cuando el motor se detiene, el par de retención será aproximadamente 1/10 de la configuración de corriente seleccionada [15].



Figura 14 Driver TB6600

## 2.10 Llave de paso

La llave de paso es un componente esencial en cualquier sistema hidráulico, ya que desempeña un papel crucial en el control del flujo de agua en una tubería. Se encuentran disponibles diversos tipos de llaves de paso, las cuales se distinguen principalmente por su método de apertura y el tipo de conexión que ofrecen. Dentro de este espectro, las llaves de paso empotradas destacan como elementos indispensables en las instalaciones de fontanería residencial y comercial,

permitiendo la regulación precisa del flujo de agua en distintos espacios. La variedad de modelos de llaves de paso empotradas ofrece opciones adaptadas a diversas necesidades y aplicaciones específicas. En lo que respecta al mecanismo de apertura, las llaves de paso con maneta de palanca sobresalen como las más habituales y son empleadas en la mayoría de las instalaciones. Estas llaves se accionan girando la maneta en sentido horario o antihorario para abrir o cerrar el paso del agua. Por otro lado, las llaves de paso con apertura de mariposa presentan una palanca que se desplaza lateralmente para controlar el flujo de agua, ofreciendo una alternativa funcional y práctica para diversas configuraciones hidráulicas [16].

## **2.11 Raspberry Pi4**

Raspberry Pi es un extraordinario dispositivo que condensa todas las funcionalidades de una computadora completa en un formato compacto y asequible (figura 15). Conocida como una computadora de placa única, esta innovación tecnológica representa una revolución en el mundo de la informática, al reunir en un solo circuito impreso todas las capacidades de una computadora tradicional (ver anexo 5).

Aunque su tamaño sea reducido, su potencial es inmenso, equiparándose en rendimiento a las computadoras de escritorio, portátiles o incluso a los smartphones más avanzados. Esta singularidad de diseño ofrece una versatilidad sin precedentes, permitiendo que Raspberry Pi sea utilizado en una amplia variedad de aplicaciones, desde proyectos de bricolaje hasta sistemas de control industrial, educación y mucho más. Su accesibilidad y bajo costo lo convierten en una opción ideal para quienes buscan explorar y desarrollar soluciones tecnológicas innovadoras sin comprometer la calidad ni la capacidad [17].

Partes de la raspberrypi4

- A. Entrada de alimentación USB tipo C
- B. Puerto de visualización DSI
- C. Inalámbrico/Bluetooth
- D. Micro HDMI 0

- E. Micro HDMI 1
- F. Sistema en chip
- G. GPIO
- H. RAM
- I. Puerto de cámara CSI
- J. AV de 3,5 mm
- K. PoE
- L. USB 2.0
- M. USB 3.0
- N. Puerto Ethernet



*Figura 15 Raspberry Pi.*



Situado sobre los puertos HDMI se halla un puerto de alimentación USB tipo C, figura 16, empleado para conectar la Raspberry Pi a una fuente de energía. Este tipo de puerto USB es frecuente en smartphones, tabletas y otros dispositivos móviles. Aunque es posible utilizar un cargador de móvil convencional para alimentar la Raspberry Pi, se recomienda utilizar la fuente de alimentación oficial de Raspberry Pi tipo C para obtener resultados óptimos.



*Figura 16 Puerto de alimentación.*

En la parte superior de la placa se encuentra un conector de pantalla, conocido como Display Serial Interface (DSI), mostrado en la figura 17, diseñado específicamente para su uso con una pantalla táctil de Raspberry Pi.

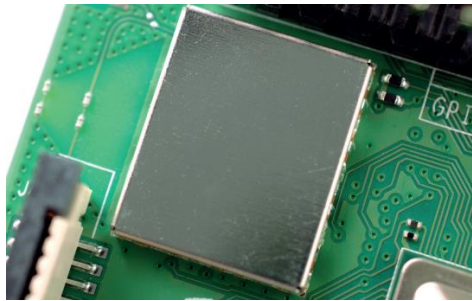


*Figura 17 Puerto de visualización DSI*

En la figura 18 se muestra la esquina superior derecha del tablero, donde se localiza una cubierta metálica que protege la radio, un elemento crucial que permite a la Raspberry Pi establecer comunicación inalámbrica con otros dispositivos.

Esta radio funciona en dos capacidades principales: como una conexión WiFi para acceder a redes informáticas, y como una conexión Bluetooth para vincularse con periféricos como ratones, así

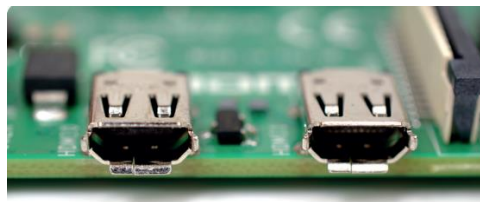
como para intercambiar datos con dispositivos inteligentes cercanos, como sensores y teléfonos móviles.



*Figura 18 Inalámbrico/Bluetooth*

Encima de eso, dentro de la figura 19 se muestra el borde izquierdo de la placa, donde se encuentran los puertos micro HDMI, que son una variante más compacta de los conectores HDMI que se encuentran comúnmente en consolas de juegos, decodificadores y televisores.

La parte "multimedia" de su nombre indica que pueden transmitir tanto señales de audio como de vídeo, mientras que "alta definición" sugiere una calidad de figura excelente. Estos puertos se utilizan para conectar la Raspberry Pi a uno o dos dispositivos de visualización, como un monitor de computadora, un televisor o un proyector.



*Figura 19 Micro HDMI 0*

Raspberry Pi, como cualquier otra computadora, está conformada por diversos componentes, cada uno con su función específica para garantizar su funcionamiento. Entre estos elementos, uno de los más cruciales, ubicado en la parte superior del tablero y protegido por una cubierta metálica, ilustrado en la figura 20, es el sistema en chip (SoC), el cual es un sistema informático

en miniatura capaz de integrar múltiples componentes, como procesadores, memoria e interfaces de entrada/salida, en un solo chip.



*Figura 20 Sistema en chip.*

En la figura 21 se muestra el borde derecho de la placa, donde se encuentra un conjunto de 40 pines metálicos, distribuidos en dos filas de 20 pines. Este es el encabezado GPIO (entrada/salida de uso general), una característica fundamental de Raspberry Pi que se emplea para establecer comunicación con hardware adicional, que va desde simples LED y botones hasta sofisticados sensores de temperatura.



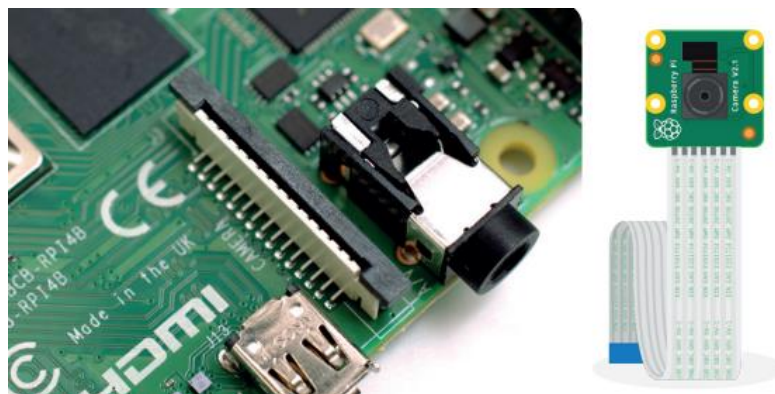
*Figura 21 GPIO*

En la figura 22 se puede observar la memoria RAM de Raspberry Pi, la cual, cuando se está utilizando el dispositivo, almacena temporalmente la información sobre las tareas que estás realizando. A diferencia de la tarjeta microSD, que es no volátil y retiene su contenido incluso cuando Raspberry Pi se apaga, la RAM volátil pierde su contenido cada vez que el dispositivo se apaga.



*Figura 22 RAM*

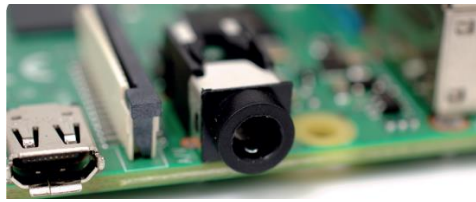
Justo sobre el conector de audio de 3,5 mm se ubica un peculiar conector con una solapa de plástico que puede levantarse; este es el conector de la cámara, también reconocido como la interfaz serie de cámara (CSI), figura 23. Este permite la conexión y uso del módulo de cámara específicamente diseñado para la Raspberry Pi, conocido como el módulo de cámara Raspberry Pi.



*Figura 23 CSIPoF*

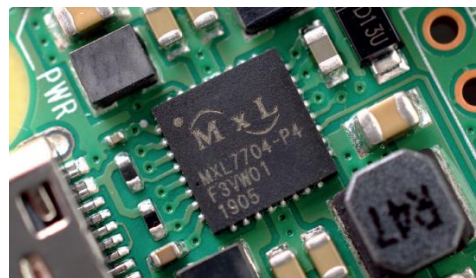
Justo por encima de los puertos USB, en el borde izquierdo de la Raspberry Pi, se encuentra un conector audiovisual (AV) de 3,5 mm (véase la Figura 24). Este conector, también conocido como conector para auriculares, se puede utilizar con ese propósito específico, aunque para una calidad de sonido óptima se recomienda conectarlo a altavoces amplificados en lugar de auriculares. Sin embargo, cuenta con una característica adicional que puede no ser tan evidente: además de transmitir audio, el conector AV de 3,5 mm transporta una señal de vídeo que puede conectarse

a televisores, proyectores y otras pantallas que admitan una señal de vídeo compuesto mediante un cable especial conocido como tip-ring-ring-sleeve (TRRS).



*Figura 24 Av 3.5 mm*

Se puede ver otro chip negro cubierto de plástico en el borde inferior de la placa, justo detrás del conjunto central de puertos USB. Este es el controlador USB y es responsable de ejecutar los cuatro puertos USB. Junto a este hay un chip aún más pequeño, el controlador de red, que maneja el puerto de red Ethernet de Raspberry Pi. Se puede encontrar un último chip negro, más pequeño que el resto, un poco encima del conector de alimentación USB tipo C en la parte superior izquierda de la placa esto se conoce como circuito integrado de administración de energía (PMIC) y se encarga de convertir la energía que ingresa desde el puerto micro USB en la energía que Raspberry Pi necesita para funcionar (ver figura 25).



*Figura 25 Circuito integrado de administración de energía (PMIC).*

Raspberry Pi cuenta con una variedad de puertos, destacando cuatro puertos Universal Serial Bus (USB), como se observa en la figura 26, se encuentran ubicados hacia el centro y el lado derecho del borde inferior. Estos puertos posibilitan la conexión de una amplia gama de periféricos compatibles con USB, que van desde teclados y ratones hasta cámaras digitales y unidades flash. En términos técnicos, existen dos tipos de puertos USB distinguibles por su color interno: los

puertos USB 2.0, identificados por partes negras en su interior, basados en la versión dos del estándar Universal Serial Bus; y los puertos USB 3.0, reconocibles por partes azules en su interior, que ofrecen una mayor velocidad y se basan en la versión tres más reciente del estándar.



*Figura 26 Puerto USB 2.0*

Justo al lado de los puertos USB se encuentra un puerto Ethernet, comúnmente denominado puerto de red. Este puerto permite la conexión de la Raspberry Pi a una red informática cableada mediante un cable que tiene un conector RJ45 (ver figura 27) en uno de sus extremos. Si se examina de cerca el puerto Ethernet, se pueden identificar dos diodos emisores de luz (LED) en la parte inferior; estos LEDs de estado indican el funcionamiento de la conexión.



*Figura 27 Puerto Ethernet.*

Justo debajo y ligeramente hacia la izquierda de este encabezado, se encuentra otro encabezado más pequeño que consta de cuatro pines. Este se utiliza para conectar el HAT Power over Ethernet (PoE), un accesorio opcional que permite que la Raspberry Pi reciba energía a través de una conexión de red en lugar del puerto USB tipo C.

Existe un puerto más en la Raspberry Pi, pero no está ubicado en el borde superior. Al voltear la placa, se encuentra un conector para tarjeta microSD en el lado opuesto de la placa, ilustrado en

la figura 28, opuesto al conector de la pantalla. Este es el medio de almacenamiento de la Raspberry Pi: la tarjeta microSD insertada aquí contiene todos los archivos que guardas, todo el software que instalas y el sistema operativo que permite el funcionamiento de la Raspberry Pi.



*Figura 28 MicroSD.*

Elementos necesarios para poder utilizar la Raspberry Pi4:

- Fuente de alimentación USB 5 V

Fuente de alimentación nominal de 3 amperios (3 A) y con un USB Conector tipo C. La fuente de alimentación recomendada para la Raspberry Pi es la original, ya que esta tiene las características necesarias para la demanda energética de la Raspberry Pi.

- tarjeta microSD con NOOBS

La tarjeta microSD cumple la función de memoria permanente para la Raspberry Pi. En ella se almacenan todos los archivos creados, el software instalado y el propio sistema operativo. Inicialmente, se suele utilizar una tarjeta de 8GB, aunque optar por una de 16GB ofrece más espacio para futuras expansiones. Utilizar una tarjeta con NOOBS (New Out Of Box Software) preinstalado puede ahorrar tiempo en el proceso de configuración.

- teclado y ratón USB

Tanto el teclado como el ratón brindan control sobre la Raspberry Pi. Prácticamente cualquier teclado o ratón, ya sea con cable o inalámbrico y con conector USB, será compatible con Raspberry Pi.



- Cable micro HDMI

Esto lleva sonido e imágenes desde Raspberry Pi a su televisor o monitor. Un extremo del cable tiene un conector micro HDMI para Raspberry Pi; el otro, un conector HDMI de tamaño completo para su pantalla. O puede utilizar un adaptador micro-HDMI a HDMI y un cable HDMI estándar de tamaño completo. Si usa un monitor

Sin una toma HDMI, puedes comprar adaptadores micro HDMI a DVI-D, DisplayPort o VGA. Para conectarse a un televisor antiguo que usa video compuesto o tiene una toma SCART, use un cable de audio/video con punta, anillo, manga (TRRS) de 3,5 mm.

## Lenguajes de programación para Raspberry pi4

Utilizar Raspberry Pi no se limita simplemente a emplear software desarrollado por otros; se trata de dar forma a tu propio software, inspirándote en cualquier idea que tu imaginación pueda concebir. Ya tengas experiencia previa en la creación de tus propios programas, un proceso conocido como programación o codificación, o no, descubrirás que Raspberry Pi es una plataforma excepcional para fomentar la creatividad y la experimentación.

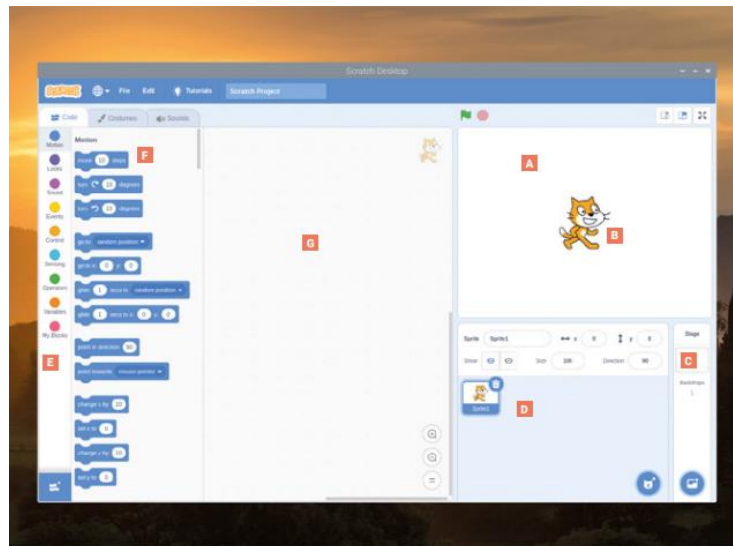
### **2.11.1 Programación con Scratch.**

La clave para hacer que la programación sea accesible en Raspberry Pi es Scratch, una herramienta visual desarrollada por el Instituto Tecnológico de Massachusetts (MIT). A diferencia de los lenguajes de programación convencionales que requieren escribir instrucciones en texto para que la computadora las ejecute, Scratch permite construir programas paso a paso utilizando bloques: piezas de código predefinidas que se ocultan detrás de fragmentos de rompecabezas codificados por colores, como se puede observar en la figura 29.

Scratch sirve como un excelente punto de partida para programadores de todas las edades, ya que ofrece una interfaz amigable. Sin embargo, es un entorno de programación potente y



completamente funcional que te permite crear desde juegos simples y animaciones hasta proyectos de robótica interactiva más complejos.



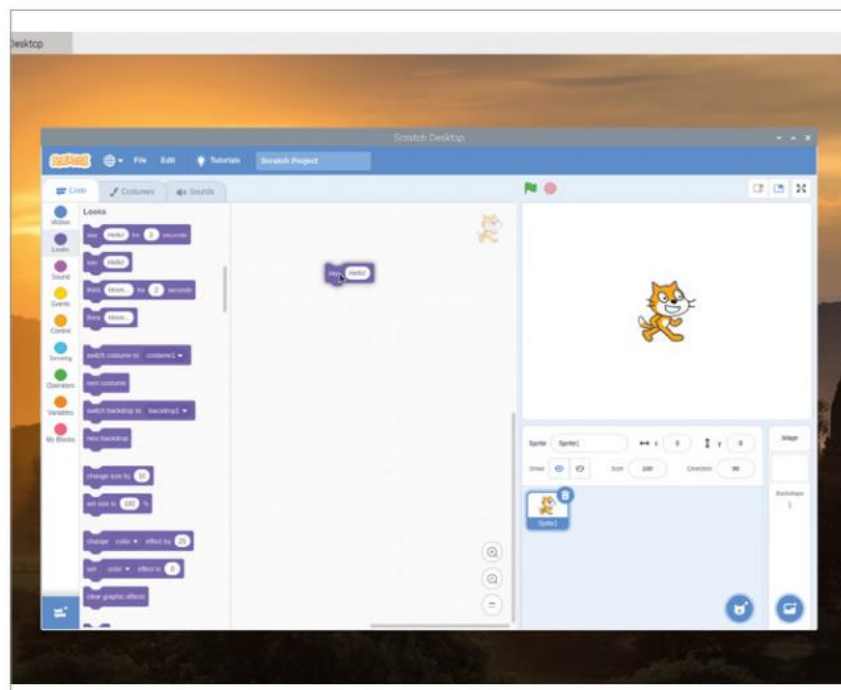
*Figura 29 Interfaz Scratch.*

- A. Zona del escenario: similar a los actores en un escenario teatral, tus sprites se mueven por esta área bajo el control de tu programa.
- B. Sprite: los personajes u objetos que manejas en un programa Scratch se denominan sprites y se sitúan en el escenario.
- C. Controles del escenario: utilizando estos controles, puedes modificar el escenario, incluyendo la opción de agregar tus propias imágenes como fondo.
- D. Lista de sprites: aquí encontrarás todos los sprites que hayas creado o importado en Scratch, mostrados en esta sección de la ventana.
- E. Paleta de bloques: en esta paleta, se encuentran todos los bloques disponibles para tu programa, organizados en categorías codificadas por colores.
- F. Bloques: estos son fragmentos de código de programa predefinidos que te permiten construir tu programa paso a paso.
- G. Área de código: es el espacio donde construyes tu programa arrastrando y soltando bloques desde la paleta de bloques para formar scripts.

Tu primer programa Scratch: ¡Hola mundo!

Para abrir Scratch 3 en Raspberry Pi, simplemente haz clic en el ícono de Raspberry Pi para acceder al menú del sistema operativo. Luego, desplázate hasta la sección "Programación" y selecciona "Scratch 3". Tras unos instantes, se abrirá la interfaz de usuario de Scratch 3. A diferencia de la mayoría de los lenguajes de programación que requieren instrucciones escritas para indicarle a la computadora qué hacer, Scratch opera de manera diferente.

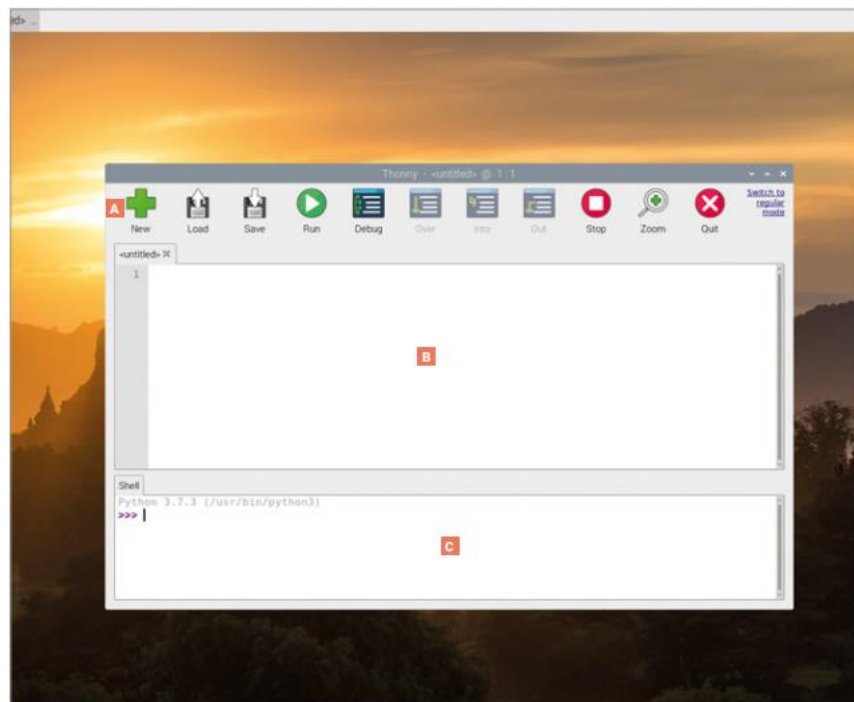
Para comenzar a utilizar Scratch, selecciona la categoría "Apariencias" en la paleta de bloques ubicada a la izquierda de la ventana de Scratch. Esta categoría se distingue por su color violeta. Busca el bloque que contiene el mensaje "¡Hola!", luego haz clic y mantén presionado el botón izquierdo del ratón sobre él. Arrastra este bloque hacia el área de código en el centro de la ventana de Scratch y suéltalo para agregarlo como se muestra en la figura 30. De esta forma se pueden desarrollar softwares aún más complejos.



*Figura 30 Ventana de Scratch*

### 2.11.2 Programación con Python

Python es un lenguaje de programación versátil que impulsa una amplia gama de proyectos. A diferencia del entorno visual de Scratch, Python se basa en texto: aquí, escribes instrucciones utilizando un lenguaje simplificado y un formato específico que la computadora luego ejecuta. Python representa un excelente paso adelante para aquellos que ya están familiarizados con Scratch, ya que proporciona una mayor flexibilidad y un entorno de programación más convencional. No obstante, esto no significa que sea difícil de aprender: con un poco de práctica, cualquier persona puede escribir programas en Python para realizar desde simples cálculos hasta proyectos sorprendentemente complejos. Existen diversos entornos de desarrollo que nos permiten programar en Python, pero en la Raspberry pi se puede utilizar el entorno Thonny mostrado en la figura 31



*Figura 31 IDE de Python.*

- A. En la interfaz C 'Modo simple' de Thonny, la barra de herramientas está compuesta por iconos amigables que funcionan como un menú, permitiéndole crear, guardar, cargar y ejecutar sus programas Python, así como realizar pruebas de varias formas.

- B. El área de secuencia de comandos es el espacio donde se redactan los programas Python. Está dividida en un área principal para escribir el código y un margen lateral más pequeño que muestra los números de línea.
- C. El Python Shell le posibilita escribir instrucciones individuales que se ejecutan instantáneamente al presionar la tecla ENTER. Además, proporciona información sobre los programas en ejecución.

La Raspberry contiene programas preinstalados en Raspberry Pi, Thonny se encuentra disponible en el menú: simplemente haz clic en el icono de frambuesa, mueve el cursor a la sección de Programación y selecciona Thonny Python IDE.

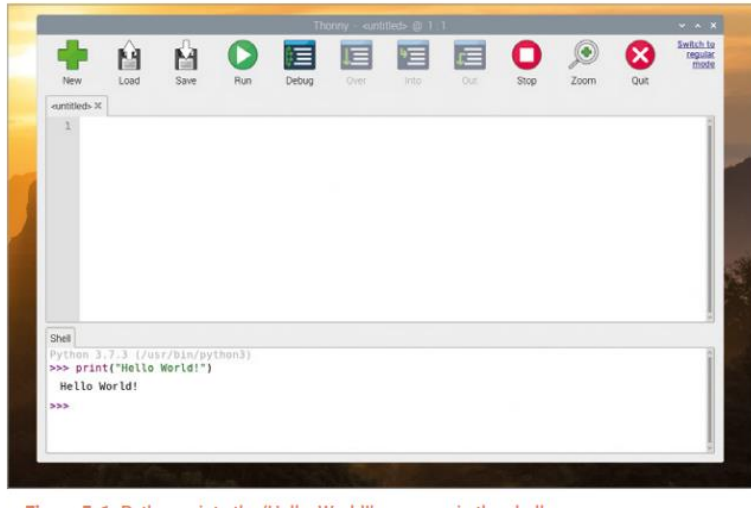
Después de unos segundos, se abrirá la interfaz de usuario de Thonny, que por defecto está en modo simple.

Thonny es un tipo de software conocido como entorno de desarrollo integrado (IDE), que, a pesar de su nombre complicado, en realidad ofrece una solución sencilla: reúne todas las herramientas necesarias para escribir o desarrollar software en una única interfaz de usuario.

Existen diversos IDE disponibles, algunos son compatibles con múltiples lenguajes de programación, mientras que otros, como Thonny, están diseñados para ser específicos de un solo lenguaje.

A diferencia de Scratch, que proporciona bloques de construcción visuales para la creación de programas, Python es un lenguaje de programación más convencional en el que todo se escribe. Para comenzar con tu primer programa en Python, simplemente haz clic en el área del shell de Python en la parte inferior de la ventana de Thonny, luego escribe la siguiente instrucción antes de presionar la tecla ENTER: ```print("¡Hola, mundo!")```

Al presionar ENTER, verás que tu programa se ejecuta de inmediato: Python responderá en la misma área del shell con el mensaje '¡Hola, mundo!', tal como lo solicitaste, figura 32.



*Figura 32 Programa ejecutado.*

### **2.11.3 Programación en Java**

Java es un lenguaje de programación a la par es una plataforma informática que apareció por primera vez en primera vez en 1995 por Sun Microsystems (posteriormente adquirida por Oracle en 2010). Inicialmente, fue diseñado para la creación de programas dirigidos a decodificadores y dispositivos portátiles, pero luego se convirtió en una opción muy popular para el desarrollo de aplicaciones web. Al ser un lenguaje de programación multiplataforma y multiparadigma, está presente en numerosos dispositivos en todo el mundo y es utilizado por aproximadamente 6,8 millones de desarrolladores.

De acuerdo con la explicación de TechTerms, Java es un lenguaje de programación orientado a objetos, lo que significa que la mayoría de los programas Java contienen "clases", que describen uno o más objetos y sirven como plantilla para crear o instanciar objetos específicos dentro de un programa. Java se caracteriza por su rigurosidad, lo que implica que las variables y funciones deben definirse explícitamente, lo que puede ayudar a prevenir errores o excepciones, aunque también puede limitar otros tipos de errores causados por variables no definidas o tipos no asignados.

El principal objetivo de Java es permitir que los desarrolladores escriban código una vez y que este pueda ejecutarse en cualquier plataforma, un concepto conocido como WORA (Write Once, Run Anywhere / Escribe una vez, ejecuta en cualquier lugar). Para que el código compilado pueda ejecutarse en cualquier plataforma que admita el lenguaje de programación Java sin necesidad de ser recompilado.

Java fue creado con cinco objetivos principales que definen su filosofía:

- Utilizada específicamente para realizar programas orientados a objetos.
- Ejecución de un programa en diferentes sistemas operativos.
- Incluir soporte integrado para trabajo en red.
- Ejecutar los códigos en sistemas remotos de manera segura.
- Ser fácil de usar y adoptar lo mejor de otros lenguajes orientados a objetos, como C++.

Actualmente, según el índice TIOBE, Java es el segundo lenguaje de programación más utilizado en el mundo, solo superado por C. Java está disponible para su descarga gratuita y [18].

#### **2.11.4 Programación en C++**

El lenguaje C++ (figura 33) ocupa una posición de gran relevancia en la industria del software debido a una serie de características fundamentales que lo distinguen por su eficacia, flexibilidad, portabilidad y su sólido respaldo para la programación orientada a objetos [19].

Se trata de un lenguaje altamente versátil y poderoso, lo que lo convierte en una opción sumamente popular para una amplia variedad de aplicaciones de software en diferentes ámbitos industriales. Desde el desarrollo de sistemas operativos hasta la creación de aplicaciones empresariales y de entretenimiento, C++ ofrece una amplia gama de posibilidades.

Entre los pilares que sustentan su gran impacto en el panorama tecnológico, destacan los siguientes aspectos:

1. Eficiencia y rendimiento: La capacidad de C++ para trabajar a bajo nivel lo convierte en una elección ideal para aplicaciones que demandan un alto rendimiento y una utilización eficiente de los recursos del sistema. Ejemplos de ello incluyen sistemas operativos, motores de juegos y aplicaciones de procesamiento de datos intensivos.
2. Flexibilidad y portabilidad: C++ es un lenguaje portátil que puede ejecutarse en una amplia variedad de plataformas y sistemas operativos, lo que lo hace extremadamente adaptable para el desarrollo de software multiplataforma.
3. Amplia adopción en la industria: Con una sólida presencia en la industria del software, C++ cuenta con una extensa base de usuarios y una abundancia de bibliotecas de código abierto disponibles. Esto facilita el proceso de desarrollo de software y promueve la colaboración entre desarrolladores.
4. Soporte para la programación orientada a objetos: C++ es un lenguaje que se fundamenta en el paradigma de la programación orientada a objetos. Estas características facilitan la creación de software modular, reutilizable y de fácil mantenimiento.



*Figura 33 Logo de C++*

#### **2.11.5 Programación en HTML**

La primera vez que se introdujo el lenguaje de programación para convertir HTML en un estándar se realizó en 1993 por parte del organismo IETF. Aunque se consiguen avances significativos, no obstante, no se logró convertir en estándar.

HTML es el lenguaje de programación que utiliza el código en etiquetas definiendo los elementos que contiene la página web que incluye los siguientes elementos: texto, imágenes, etc. Dichas etiquetas serán interpretadas por un programa navegador de internet.

Un documento HTML en realidad es un archivo de texto, por lo cual para crear o modificar archivos HTML se puede utilizar un editor de texto simple, existen diferentes como lo son: FrontPage o Dreamweaver, estos editores presentan el código de forma más amigable, distinguiendo entre los diferentes tipos de etiquetas con colores diversos y comparando herramientas visuales para más elementos.

### **El código HTML.**

La especificación del código en su última versión de HTML 4.0 se puede encontrar en su página web gratuita, esta norma oficial está escrita de manera muy formal y algunas secciones son difíciles de comprender.

Este lenguaje se basa en marcas o etiquetas.

Por lo tanto, los elementos que conforman la página web se describen mediante palabras especiales que marcan el inicio y el final de estos, estas palabras se llaman etiquetas.

Dichas etiquetas se indican por pares como se muestra:

Etiqueta de apertura: carácter <, nombre de la etiqueta (es importante no dejar algún espacio) y terminando con el carácter >.

Etiqueta de cierre: carácter <, seguido del nombre de la etiqueta (sin ningún espacio en el nombre) y terminar con el carácter >.

### **Estructura de un documento HTML.**

Todas las páginas contienen los mismos elementos



Doctype: Primera línea del código que tiene cualquier documento HTML. Esta indica al navegador qué especificación de HTML se utiliza. Suele ser como se muestra en la figura 34.



*Figura 34 Primera línea de HTML*

HTML, el lenguaje de marcado fundamental para la creación de páginas web, utiliza diversas etiquetas para estructurar y definir el contenido de un documento. A continuación, se describen algunas de las etiquetas más comunes y su función en la construcción de páginas web:

1. Etiqueta `<html>`: Esta etiqueta, junto con su contraparte de cierre `</html>`, encierra todo el contenido del documento HTML. Sirve como el contenedor principal y proporciona un punto de inicio y finalización para el análisis del texto sin formato por parte de las aplicaciones.
2. Etiqueta `<head>`: Delimita el contenido de la cabecera del documento, que incluye elementos como el título de la página y otra información que no se muestra en la pantalla, como metadatos.
3. Etiqueta `<title>`: Rodea el texto del título de la página. Este título es visible en la barra de título del navegador web cuando se presenta la página, proporcionando una identificación rápida y concisa del contenido.
4. Etiqueta `<meta>`: Utilizada para proporcionar metainformación al documento, que puede ser aprovechada por los motores de búsqueda para indexar y clasificar el contenido de la página de manera más efectiva.
5. Etiqueta `<body>`: Encierra el contenido visible de la página, incluyendo texto, imágenes, enlaces y otros elementos. Esta etiqueta permite definir propiedades específicas de la página, como colores de fondo o estilos de texto, que afectan la apariencia visual del contenido.

Estas etiquetas son fundamentales para estructurar y organizar el contenido de una página web de manera efectiva, permitiendo una presentación coherente y accesible para los usuarios finales. Al comprender el propósito y la función de cada etiqueta, los desarrolladores web pueden crear sitios web bien diseñados y optimizados para su uso en diferentes plataformas y dispositivos [20].

## **2.12 Pantalla táctil.**

Las pantallas táctiles son dispositivos de interfaz que permiten la interacción directa tocándolas. Hay dos tipos principales de pantallas táctiles: resistivas y capacitivas, cada una con sus propias características y usos específicos.

Una pantalla táctil resistiva es aquella que resiste el contacto del dedo o de un objeto. Para activarla, se requiere aplicar cierta presión, ya sea con un lápiz óptico o electrónico, o simplemente con la fuerza del dedo. Este tipo de pantalla no responde al simple roce o deslizamiento de la mano sobre su superficie. Las pantallas táctiles resistivas son comunes en entornos como supermercados, donde se utilizan para recopilar firmas electrónicas al pagar la factura.

Por otro lado, las pantallas táctiles capacitivas están diseñadas específicamente para responder al toque de un dedo humano. Este tipo de pantalla es sensible al contacto eléctrico del dedo y puede detectar múltiples puntos de contacto simultáneamente. Las pantallas táctiles capacitivas son ampliamente utilizadas en dispositivos como teléfonos inteligentes y tabletas, siendo los tipos de pantalla más comunes en la electrónica de consumo debido a su alta sensibilidad y capacidad para soportar gestos multitáctiles [21].

## **2.13 Antecedentes**

### **2.13.1 Diseño y desarrollo de la ducha inteligente ¡Shower!.**

La tesis titulada “Diseño y desarrollo de la ducha inteligente ¡Shower!” nos muestra el proceso riguroso que se siguió para poder crear el dispositivo se menciona que la automatización es una herramienta puramente industrial que rodea los aspectos de vida del hombre.

La ducha contiene dos válvulas W35T para regular la temperatura deseada dado que tiene un estándar comercial usado en grifería tipo monomando, teniendo un torque de rotación, torque de apertura, ángulo de rotación de 90° ángulo de apertura de 25°.

Los motores fueron importados de Estados Unidos correspondientes a la empresa “Permanent Magnet Stepper Gear Motors” con la referencia TGM42-075-40-12V-040a-50R-LW6 dada la necesidad del torque y la reducción del acople necesaria, el motor es unipolar, voltaje nominal es de 12V, corriente por fase de 400mA, torque de detección sin alimentación, torque de salida estático. Se ajustaron los motores para la reducción de las revoluciones por minuto dados los ángulos de rotación necesarios. Los componentes necesarios se desarrollaron mediante un software para poder tener las piezas con las dimensiones exactas.

La interfaz de usuario tiene como principal función el número de elementos de interacción óptimo para que el usuario ingrese su información todo esto para evitar un panel compuesto por pulsadores, botones, selectores, perillas y confundir al usuario, la relevancia y comprensibilidad de la información presentada al usuario es decir la información del usuario que el usuario visualice será solo la necesaria y no de todo el proceso que realice el sistema, la rapidez de ajuste es rápida y sencilla los valores de la temperatura o el flujo de agua deben ser instantáneo para la seguridad del usuario previniendo quemaduras o otros inconvenientes, la jerarquización de la información las variables principales más importantes o principales deben ser los datos en poder visualizarse inmediatamente [22].

### **2.13.2 Rivus: Diseño y conceptualización de una ducha inteligente.**

Rivus es el diseño y conceptualización de una ducha inteligente que está compuesta por elementos tales como: aplicación para dispositivos móviles, sistemas de ducha inteligente con sensores, iluminación LED, sonido, panel táctil. El usuario puede modificar diferentes parámetros de configuración tales como: música, iluminación, temperatura, tiempo, presión del agua. Tiene dos formas de interacción por medio de la aplicación se pueden gestionar las cuentas, se pueden acceder a todas las funciones, a través del panel táctil es más fácil para personas mayores o

invitados que estarán un lapso corto en la casa, permite ajustar manualmente temperatura y presión del agua.

La ducha está diseñada para la población de 31-50 años con un nivel socioeconómico medio-alto que quieran ahorrar en el hogar con una conciencia ecológica.

La ducha aporta nuevas funciones que generan un gran ahorro económico y de agua, contribuyendo a la sostenibilidad medioambiental mediante el reaprovechamiento de esta, compuesta por una aplicación para dispositivos móviles y un sistema de ducha inteligente [23].

### **2.13.3 Sistema de ducha termostático, sistema inteligente de chorro.**

El sistema de ducha termostático de Nwiass brinda una experiencia de ducha de lujo con un enfoque inteligente y contemporáneo. Valorado en 48,600 pesos mexicanos, este sistema presenta un panel de pared de 23 pulgadas en negro mate, que incorpora chorros corporales y LED para crear un ambiente elegante en el baño.

En contraste, el panel de ducha Imperial Solana de acero inoxidable ofrece una opción más accesible, con un costo de 4,749 pesos mexicanos. Este panel termostático en níquel cuenta con una regadera de lujo que garantiza un control preciso de la temperatura, asegurando una experiencia de ducha cómoda y placentera.

Además, el panel cuenta con una pantalla LED inteligente que muestra la temperatura y la hora, proporcionando un toque moderno al baño. El rociador de hidromasaje integrado ofrece un masaje revitalizante, mientras que la regadera de mano ofrece versatilidad adicional para adaptarse a las preferencias individuales.

La sólida construcción de acero inoxidable en níquel garantiza durabilidad y resistencia a la corrosión, mientras que los accesorios incluidos, como el soporte de pulverización manual y la manguera de agua elástica de alta densidad de 1.5 m, añaden funcionalidad y comodidad.

Con múltiples métodos de salida de agua y un mango de palanca (monomando) para un control sencillo y conveniente, este panel de ducha ofrece una combinación ideal de estilo, funcionalidad y precio accesible para cualquier baño moderno [24].

#### **2.14.4 Juego de ducha inteligente, cabezal de ducha LED de techo.**

Con el Panel de Ducha Modelo Imperial, convierte tu rutina de ducha en un momento de indulgencia y confort. Experimenta la combinación ideal entre diseño contemporáneo y características avanzadas, mientras agregas un toque de sofisticación moderna a tu cuarto de baño [25].

#### **2.13.5 Panel de ducha de acero inoxidable modelo Imperial Solana.**

Es una regadera de lujo termostática color niquel con pantalla LED inteligente marca temperatura y hora / rociador de hidromasaje y regadera de mano.

El juego de ducha de techo con luz LED inteligente tiene un precio de 15.642 pesos mexicanos [26]. Presenta una serie de características y funciones que incluyen:

- Dimensiones: 301-400 mm
- Estilo: Contemporáneo
- Número de modelo: TTT-SS015
- Grifos termostáticos y luz LED
- Control automático de termostato para agua fría/caliente
- Tipo de cabezal de ducha: Fijo y giratorio
- Material del núcleo de la válvula: Cerámica
- Número de manijas: Otro
- Tratamiento superficial: Ennegrecido
- Acabado superficial: Acero inoxidable
- Material: Acero inoxidable 304
- Función: Lluvia, cascada, ducha de mano
- Tamaño del cabezal de ducha: 500\*360\*80mm
- Instalación: Empotrado en el techo
- Luz LED: Requiere conexión a la alimentación
- Interfaz de hilo: G1/2
- Válvula de ducha: Termostática
- Control remoto de la luz LED

- Función de mezclador de ducha con tres opciones
- Material de la válvula mezcladora de ducha: Latón y cromo

### **2.13.6 Sistema ducha digital baño inteligente indicador.**

El sistema de ducha digital indicador de baño inteligente está disponible por 8,191 pesos mexicanos [27]. Aquí están sus características:

- Es un dispositivo inteligente: No
- Altura ajustable: Sí
- Forma de la ducha: Cuadrada
- Acabado superficial: Cepillado
- Material: Latón
- Número de modelo: S129
- Material: Latón, acero inoxidable, ABS
- Tipo: Conjunto de sistema de ducha
- Cabezal de ducha: Digital
- 4 funciones:
- Teleducha con pantalla digital: Pantalla de temperatura
- Sistema de ducha: Baño, LED, Digital
- Conjunto de ducha: Termostático, Gris
- Juego de ducha para baño: Lujo
- Proceso de cepillado PVD: Moderno y elegante, con un nuevo estilo de cepillado gris para una estética de baño única.
- Cuerpo de latón fino: Construcción de alta calidad, con más material que los grifos convencionales.
- Vidrio nano desincrustante: Autolimpieza y resistencia al agua.
- Indicador LED de temperatura: Muestra la temperatura del agua en tiempo real.
- Interruptor tipo prensa: Fácil de usar y estéticamente agradable.

- Superficie de almacenamiento espaciosa: Ofrece espacio adicional para organizar artículos de baño.
- Instalación universal: Adecuado para la mayoría de las familias con control de frío y calor.
- Rociador de lluvia: Cubre todo el cuerpo con finas gotas de agua para una experiencia relajante.
- Orificio de salida de silicona: Evita obstrucciones y facilita la limpieza.
- Ducha de mano de tres funciones: Permite ajustar fácilmente el modo de salida de agua para satisfacer diversas necesidades.

## Capítulo III. Desarrollo e implementación

### 3.1 Diseño del control difuso en el software Matlab y su análogo en Python.

Para diseñar el control difuso consideramos utilizar diversos softwares conocidos para este propósito, sin embargo, en algunas clases de la licenciatura comenzamos a familiarizarnos con el entorno de Matlab, un software con diversas herramientas de programación y simulación que pueden ser aplicadas a varias ramas de la industria.

Por tal motivo y por los conocimientos previos del software, desarrollamos ahí nuestro control difuso y con sus herramientas de simulación comprobamos su funcionamiento.

Para ello fue necesario realizar los siguientes pasos En Matlab, dentro de la pantalla de inicio, en la ventana de comandos escribimos “fuzzy” y presionamos la tecla “enter” (figura 35), esto abrirá una ventana en la cual podremos establecer los parámetros de las variables lingüísticas a utilizar, sus valores y sus reglas, mostrada en la figura 36.

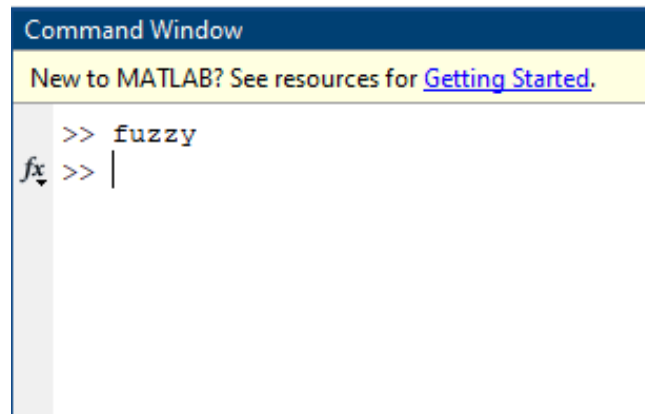
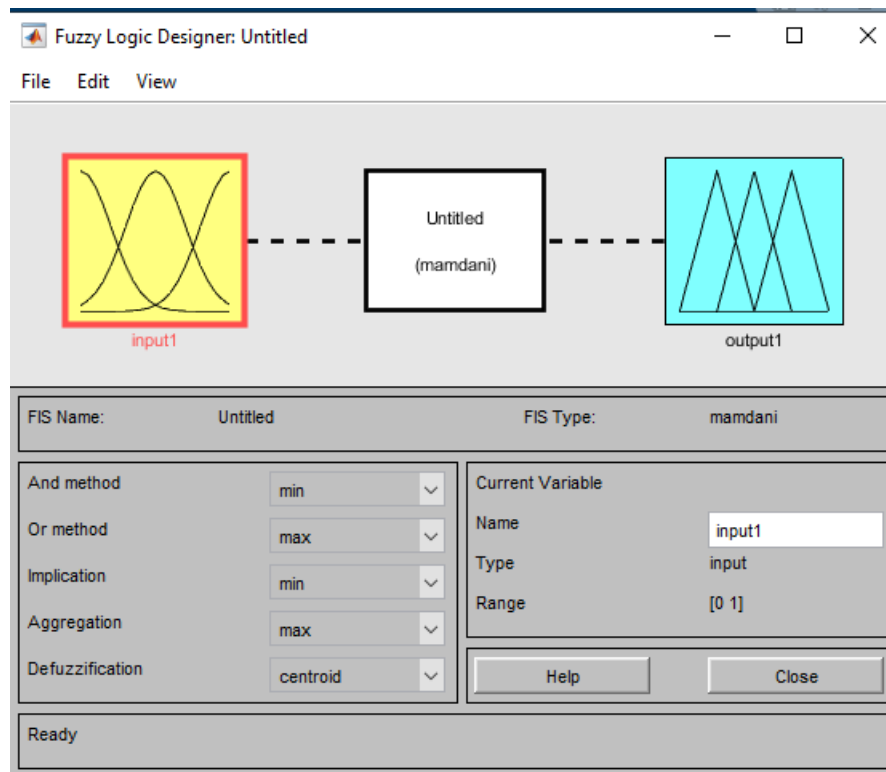


Figura 35 Ventana de comandos de Matlab con el comando fuzzy.





*Figura 36 Inicio de la ventana del diseño de lógica difusa.*

En esta ventana mencionada, se pueden cambiar los nombres de las variables a utilizar como en la y agregar las que se necesiten como en la figura 37.

Dentro de ella pueden modificarse las funciones de pertenencia de cada variable y modificar los valores que las componen como se muestra en las figuras 38 y 39.

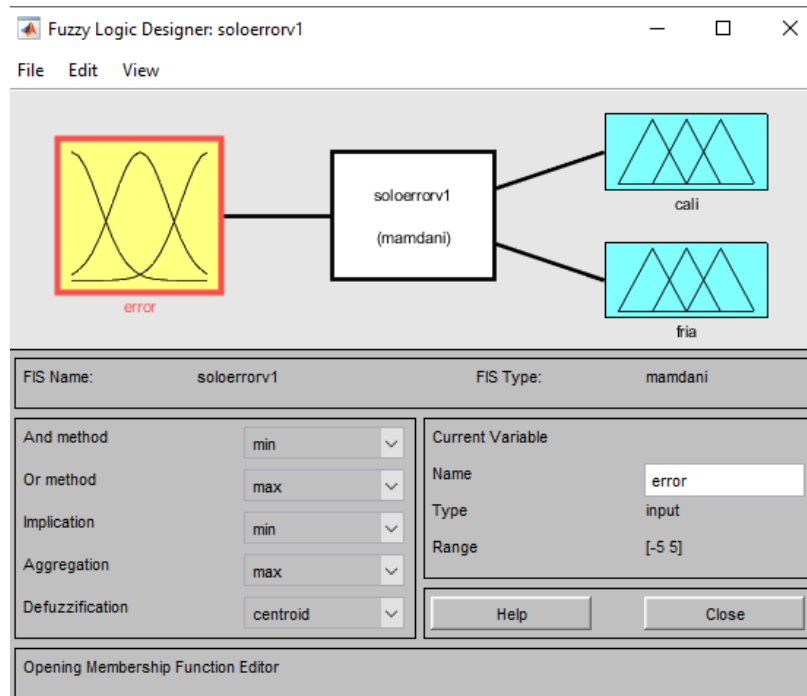


Figura 37 Asignación de variables a utilizar.

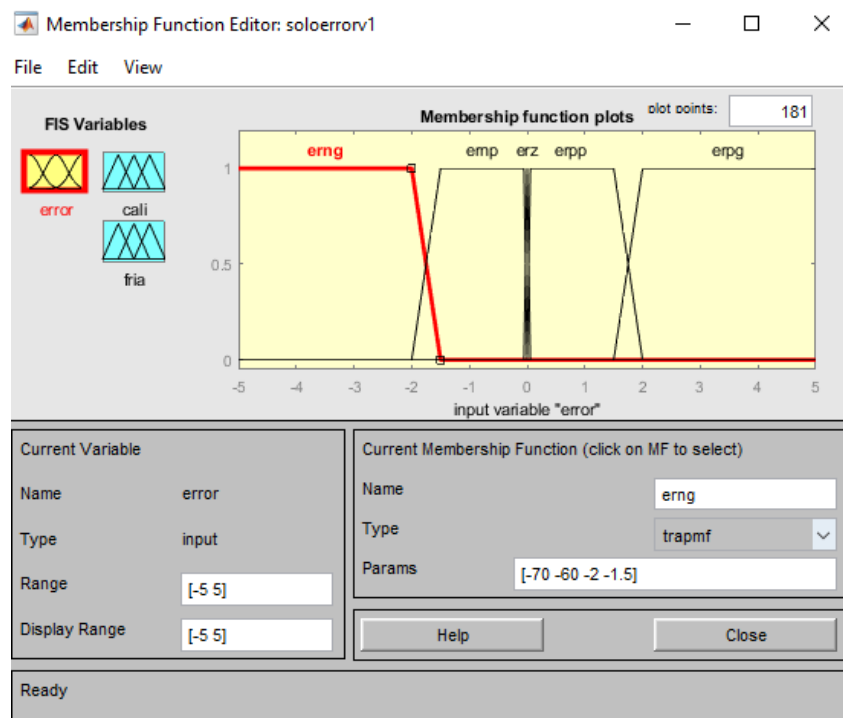
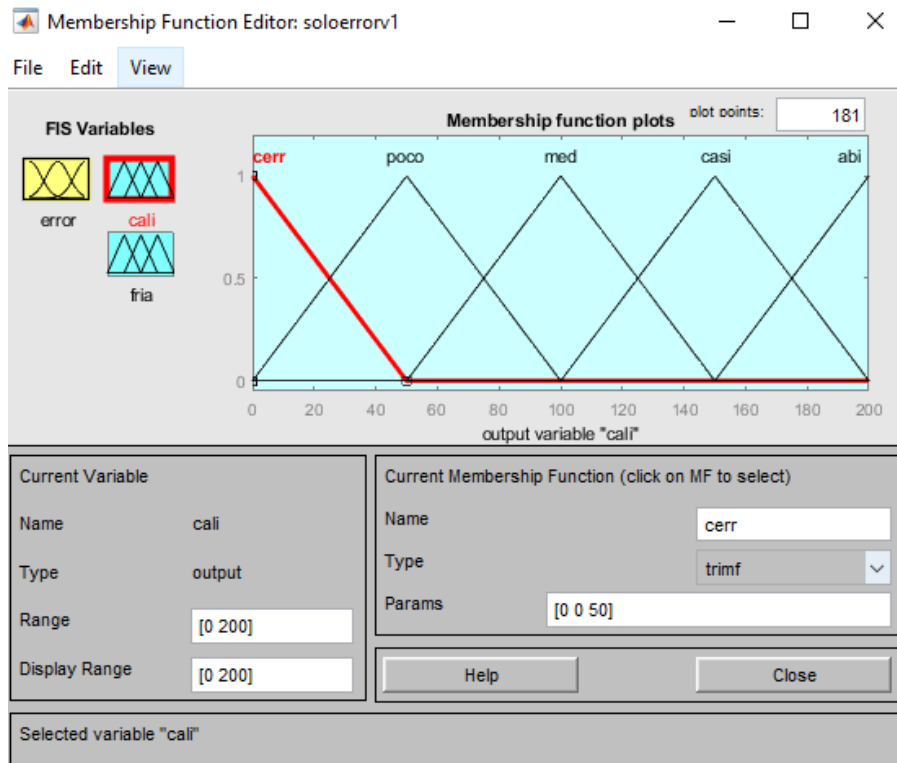


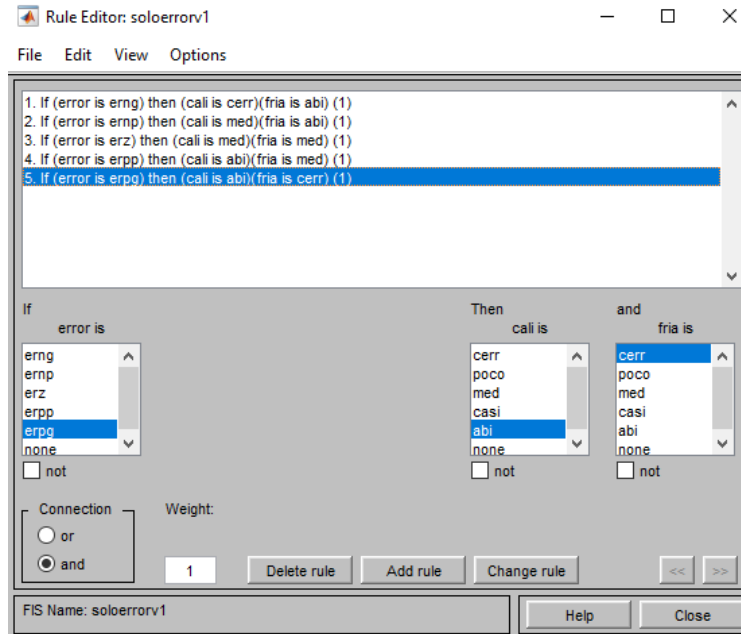
Figura 38 Visualización de la función de pertenencia de la variable error.



*Figura 39 Visualización de las funciones de pertenencia para las variables cali y fria.*

También se pueden modificar las reglas que utilizará el control difuso para que realice las regulaciones necesarias al sistema en el cual va a aplicarse, mostrado en la figura 40.

Hay aún más funciones que ofrece esta herramienta, pero las presentadas son las esenciales para el diseño del control difuso de este proyecto.



*Figura 40 Reglas establecidas en Matlab.*

Al finalizar el control difuso, es necesario exportarlo a una carpeta con una ubicación de fácil acceso y al espacio de trabajo de Matlab como se muestra en las figuras 41 a 44.

Es importante recalcar que la carpeta de trabajo de Matlab debe coincidir con la ubicación del archivo que se exportó (figura 45).

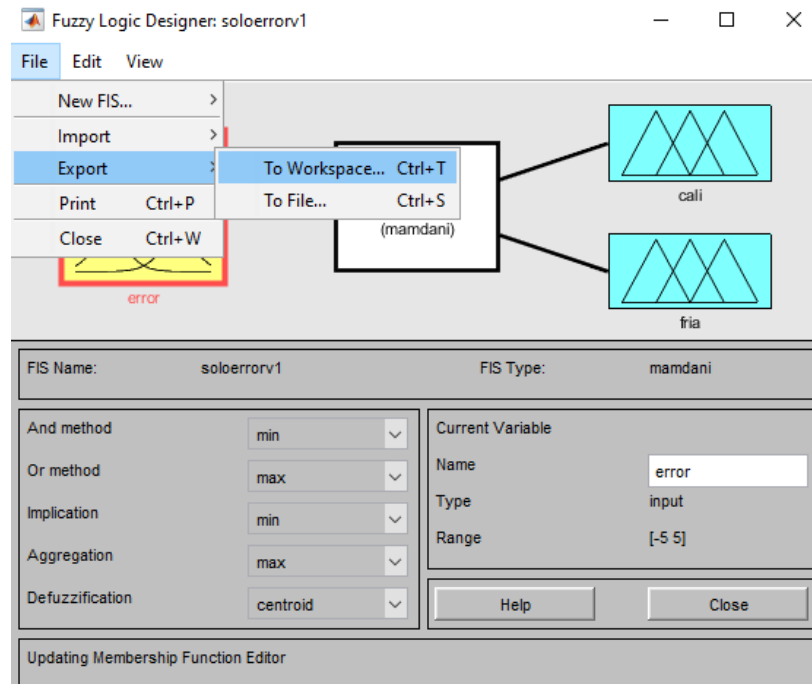


Figura 41 Exportación de archivo al espacio de trabajo (Workspace).

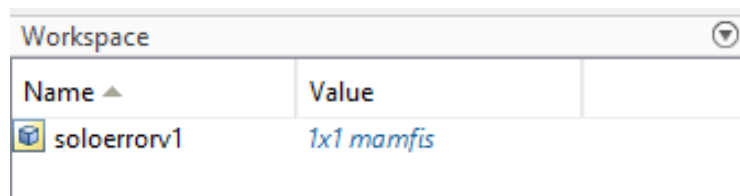


Figura 42 Workspace de Matlab con el archivo de control difuso.

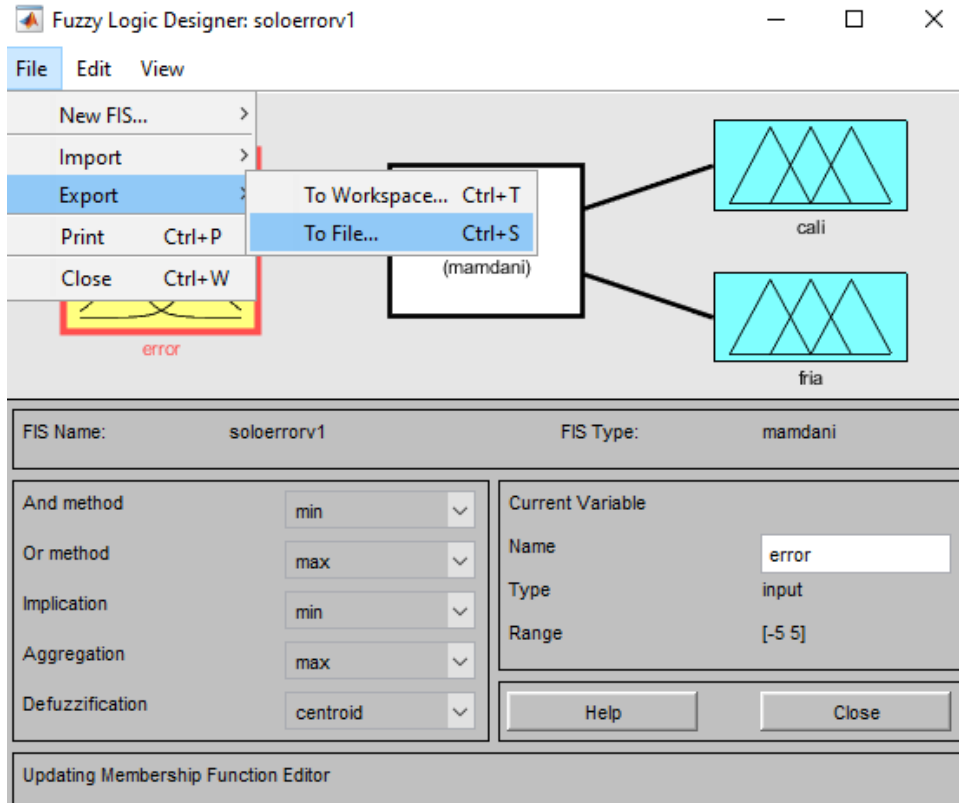


Figura 43 Exportación de archivo como documento de extensión .fis.

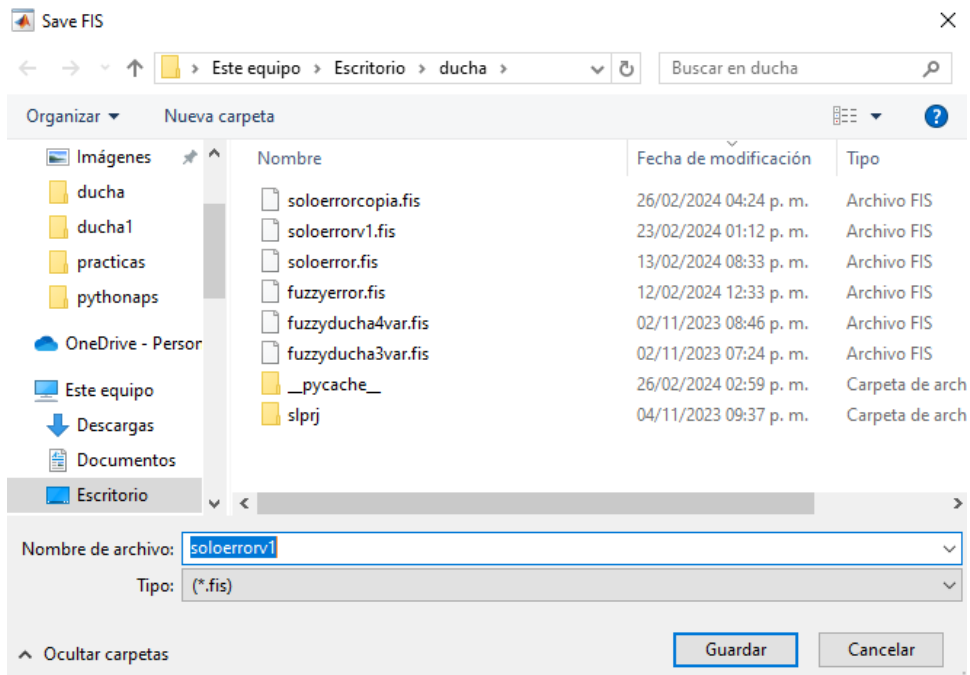


Figura 44 carpeta de guardado del archivo de control difuso.

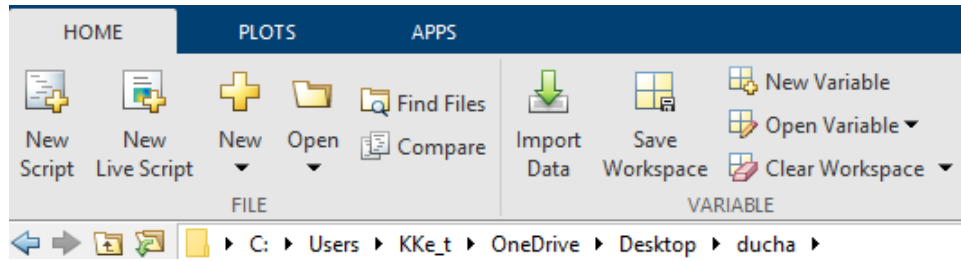


Figura 45 Cambio en el directorio de la ventana de inicio de Matlab.

Una vez listo el control difuso, es necesario hacer un esquema en Simulink que nos permita modificar los parámetros de valor\_deseado y valor\_sensado como en nuestro programa original. Para ello, accederemos a Simulink y realizaremos el esquema de la figura 46. En dicho esquema observamos las entradas de valor\_deseado y valor\_sensado, de las cuales se obtiene su diferencia, valor denominado como error y que será utilizado por el control difuso para regularse, además se observan un par de selectores tipo slider que permite variar el valor de las entradas.

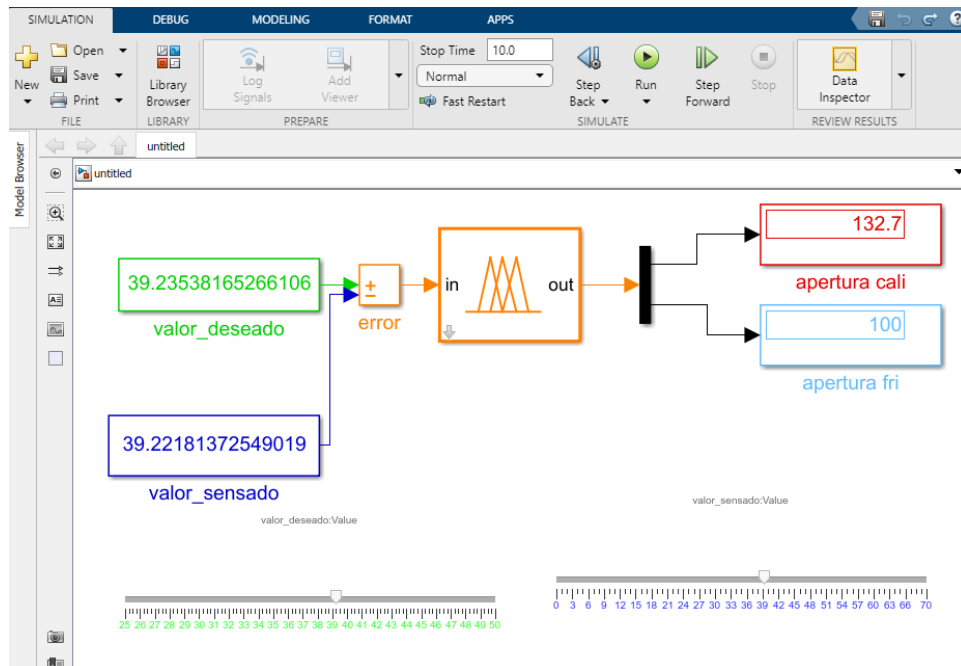
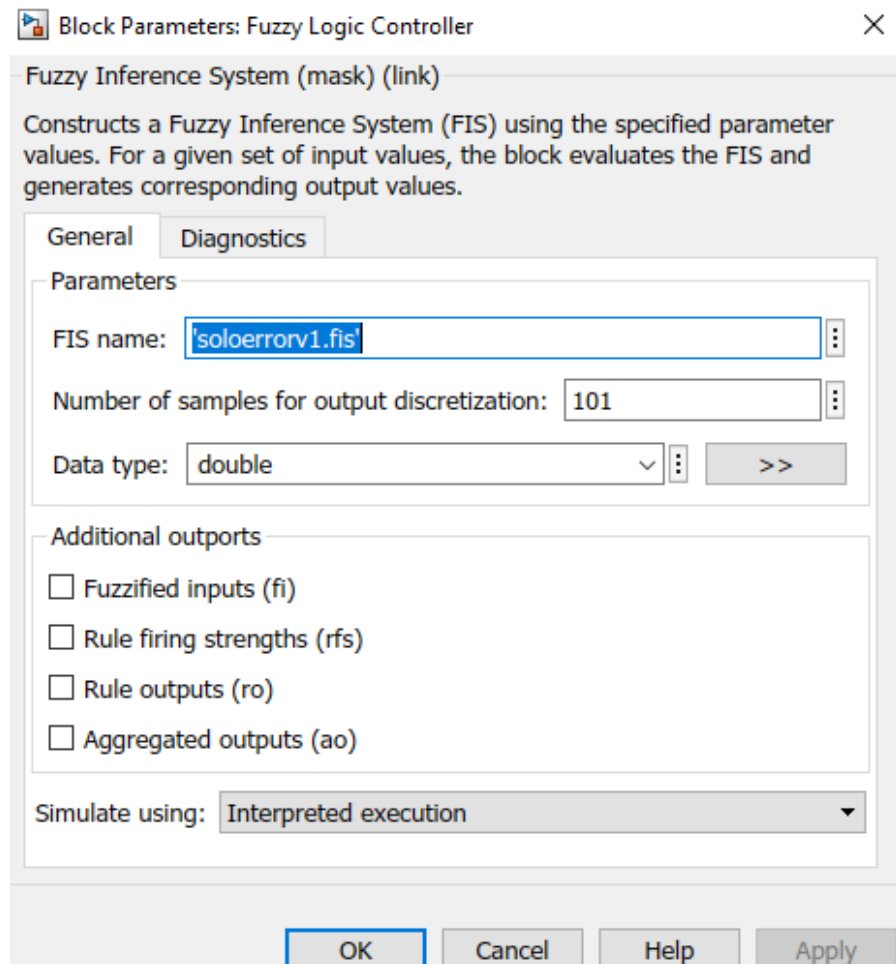


Figura 46 Esquema de Simulink para la simulación del control difuso.

Con el esquema realizado, añadiremos en el bloque de control difuso el nombre con el que guardamos nuestro archivo de control difuso (figura 47). Una vez realizado esto, puede iniciar la simulación.



*Figura 47 Asignación del archivo con extensión. fis al bloque de lógica difusa.*

Cuando el archivo está cargado en el bloque de lógica difusa, es posible iniciar las simulaciones modificando los valores de las variables de valor deseado y valor sensado haciendo uso del slider como se aprecia en las figuras 48 a la 50.



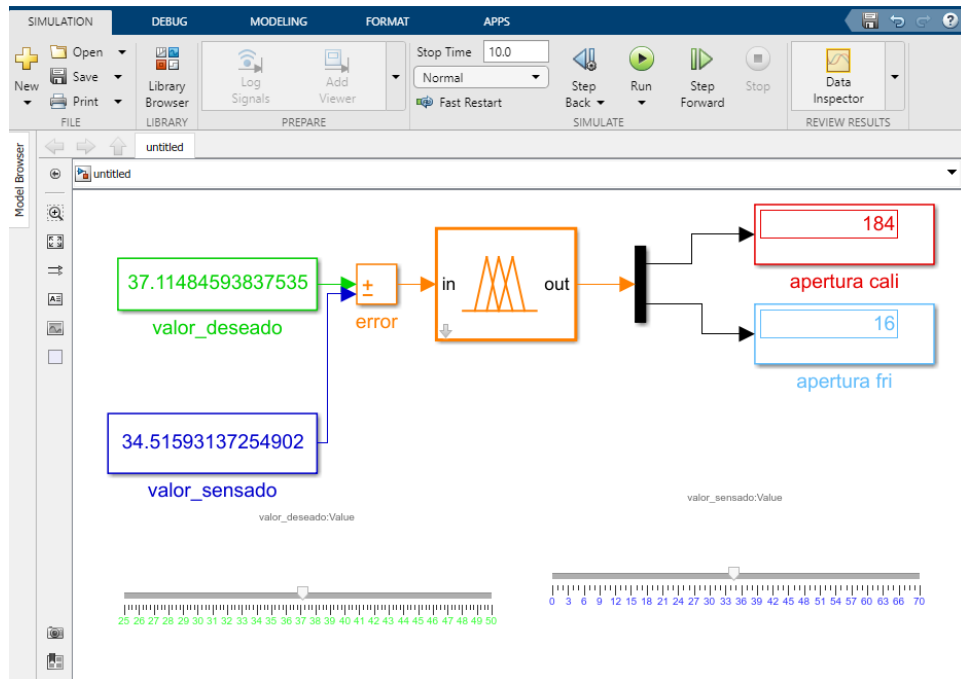


Figura 48 Simulación del control difuso con error positivo grande.

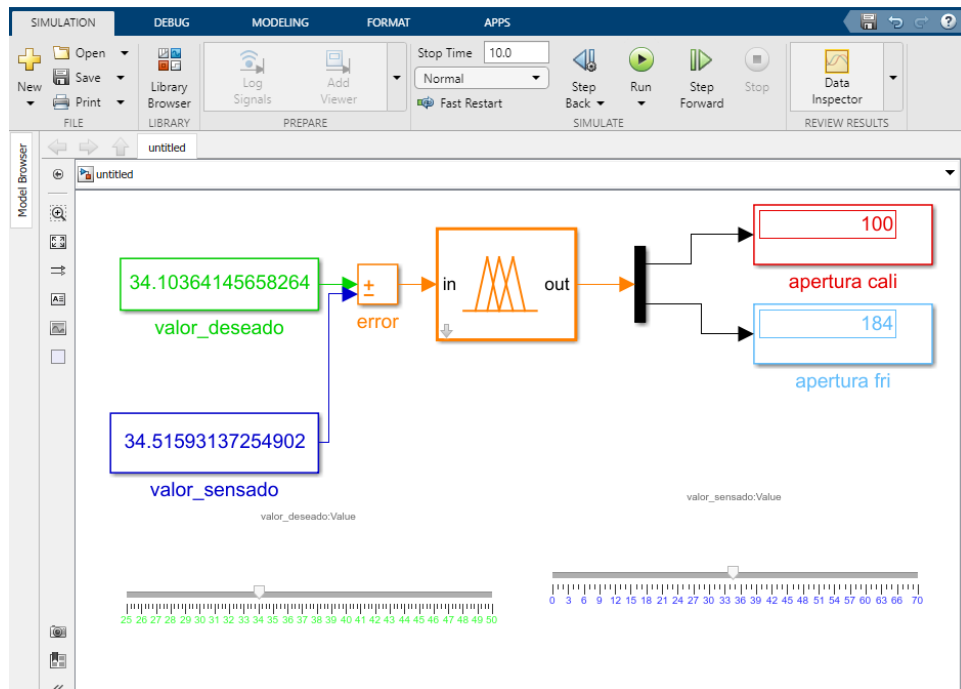
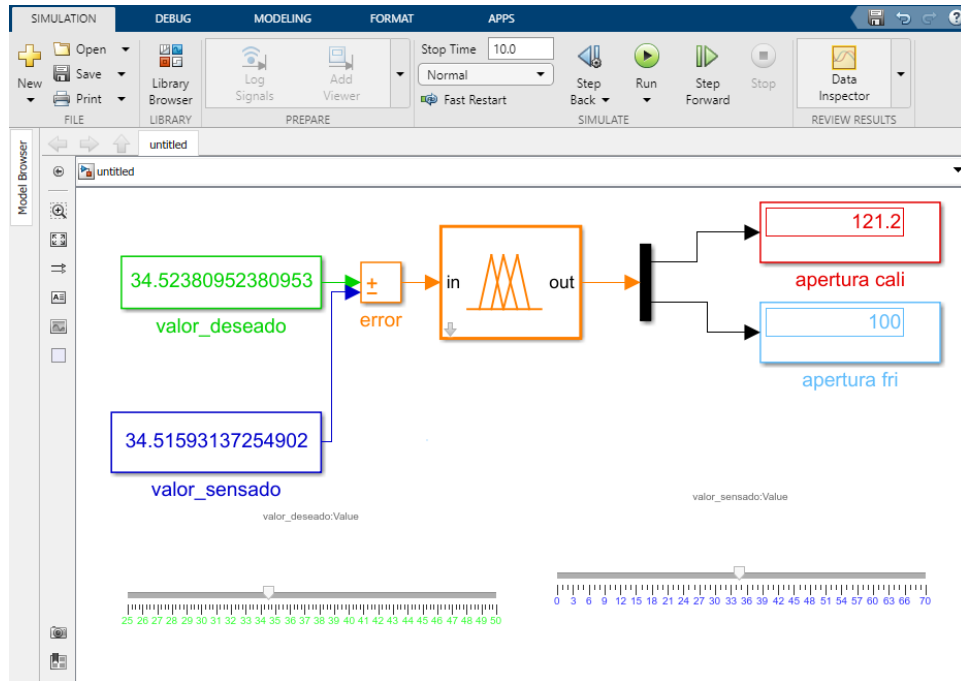


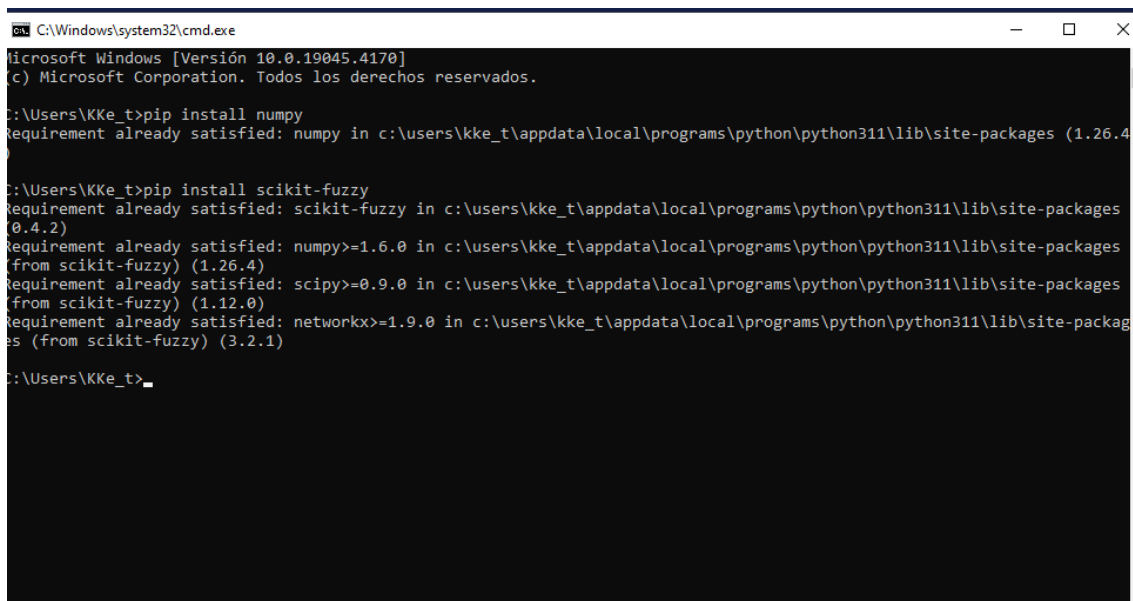
Figura 49 Simulación del control difuso con error negativo pequeño.



*Figura 50 Simulación del control difuso con error muy cercano a cero.*

El diseño del control difuso ha sido implementado dentro de una computadora personal (PC), con el fin de facilitar la edición de código y corroborar su funcionamiento. El software utilizado para tal propósito es el Visual Studio, que tiene herramientas que ayudan al programador a reconocer sus variables y errores, además de ser compatible con Python, haciendo que sea un entorno de desarrollo fácil de usar. Debido a la pantalla reducida y a problemas para conectar la Raspberry pi a un monitor externo, esta solución fue la más útil para el desarrollo del resto del proyecto en cuanto a programación. De esta forma podemos implementar los avances dentro de la Raspberry pi al copiarle el código realizado en Visual Studio. A continuación, se muestran los pasos que se siguieron para diseñar el control difuso.

Para visualizar el funcionamiento del control difuso, es necesario instalar en ambas computadoras (**Raspberry pi y en PC**) las librerías *numpy* y *scikit-fuzzy*. En la terminal de ambas computadoras, instalamos dichas librerías con los comandos “pip install numpy” y “pip install scikit-fuzzy”, como se muestra en las figuras 51 y 52.



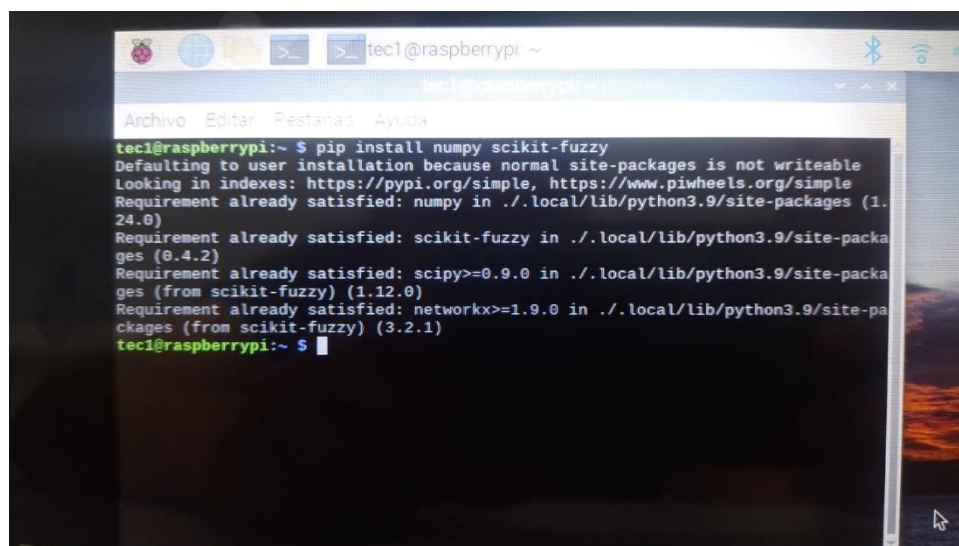
```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versión 10.0.19045.4170]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\Users\KKe_t>pip install numpy
Requirement already satisfied: numpy in c:\users\kke_t\appdata\local\programs\python\python311\lib\site-packages (1.26.4)

C:\Users\KKe_t>pip install scikit-fuzzy
Requirement already satisfied: scikit-fuzzy in c:\users\kke_t\appdata\local\programs\python\python311\lib\site-packages (0.4.2)
Requirement already satisfied: numpy>=1.6.0 in c:\users\kke_t\appdata\local\programs\python\python311\lib\site-packages (from scikit-fuzzy) (1.26.4)
Requirement already satisfied: scipy>=0.9.0 in c:\users\kke_t\appdata\local\programs\python\python311\lib\site-packages (from scikit-fuzzy) (1.12.0)
Requirement already satisfied: networkx>=1.9.0 in c:\users\kke_t\appdata\local\programs\python\python311\lib\site-packages (from scikit-fuzzy) (3.2.1)

C:\Users\KKe_t>
```

*Figura 51 Instalación de librería numpy y scikit-fuzzy.*



```
tec1@raspberrypi ~
tec1@raspberrypi ~
Archivo Editar Pestañas Ayuda
tec1@raspberrypi:~ $ pip install numpy scikit-fuzzy
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: numpy in ~/.local/lib/python3.9/site-packages (1.24.0)
Requirement already satisfied: scikit-fuzzy in ~/.local/lib/python3.9/site-packages (0.4.2)
Requirement already satisfied: scipy>=0.9.0 in ~/.local/lib/python3.9/site-packages (from scikit-fuzzy) (1.12.0)
Requirement already satisfied: networkx>=1.9.0 in ~/.local/lib/python3.9/site-packages (from scikit-fuzzy) (3.2.1)
tec1@raspberrypi:~ $
```

*Figura 52 Instalación de librerías en la Raspberry.*

Una vez instaladas las librerías, podemos integrar el código en Visual Studio, para obtener el archivo con extensión “.py” que ejecutaremos en la Raspberry. Además, dicho código puede simularse también en la laptop. En este caso es necesario introducir manualmente los valores de la temperatura sensada pues no pueden conectarse los sensores físicamente. En esta etapa de

diseño del control, solo se busca que el control difuso funcione correctamente y tener en cuenta las variables a utilizar. A continuación, en las figuras 53 a 57 se muestra el programa de control difuso.

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

class FuzzyLogicController:
    def __init__(self):
        #codigo de logica difusa
        # Variables lingüísticas
        error = ctrl.Antecedent(np.arange(-50, 50, 0.01), 'error')
        cali = ctrl.Consequent(np.arange(0, 200, 1), 'cali')
        fria = ctrl.Consequent(np.arange(0, 200, 1), 'fria')
        # Funciones de pertenencia
        self._definir_funciones_pertenencia(error, cali, fria)

        # Reglas
        self._definir_reglas(error, cali, fria)

        # Sistema de control
        self.cal_ctrl = ctrl.ControlSystem([self.rule1, self.rule2, self.rule3, self.rule4, self.rule5])
        self.cali_ctrl = ctrl.ControlSystemSimulation(self.cal_ctrl)
        self.fria_ctrl = ctrl.ControlSystem([self.rule6, self.rule7, self.rule8, self.rule9, self.rule10])
        self.fria_ctrl = ctrl.ControlSystemSimulation(self.fria_ctrl)
```

Figura 53 Código de control difuso parte 1 en Visual Studio en lenguaje Python.

```
def _definir_funciones_pertenencia(self, error, cali, fria):
    # Funciones de pertenencia
    error['erng'] = fuzz.trapmf(error.universe, [-50, -50, -2, -1.5])
    error['ernp'] = fuzz.trapmf(error.universe, [-2, -1.5, -0.02, 0])
    error['erz'] = fuzz.trimf(error.universe, [-0.02, 0, 0.02])
    error['erpp'] = fuzz.trapmf(error.universe, [0, 0.02, 1.5, 2])
    error['erpg'] = fuzz.trapmf(error.universe, [1.5, 2, 50, 50])

    cali['cerr'] = fuzz.trimf(cali.universe, [0, 0, 50])
    cali['poco'] = fuzz.trimf(cali.universe, [0, 50, 100])
    cali['med'] = fuzz.trimf(cali.universe, [50, 100, 150])
    cali['casi'] = fuzz.trimf(cali.universe, [100, 150, 200])
    cali['abi'] = fuzz.trimf(cali.universe, [150, 200, 200])

    fria['cerr'] = fuzz.trimf(fria.universe, [0, 0, 50])
    fria['poco'] = fuzz.trimf(fria.universe, [0, 50, 100])
    fria['med'] = fuzz.trimf(fria.universe, [50, 100, 150])
    fria['casi'] = fuzz.trimf(fria.universe, [100, 150, 200])
    fria['abi'] = fuzz.trimf(fria.universe, [150, 200, 200])
```

Figura 54 Código de control difuso parte 2 en Visual Studio en lenguaje Python.

```

def _definir_reglas(self, error, cali, fria):
    # Reglas
    self.rule1 = ctrl.Rule(error ['erng'], cali ['cerr'])
    self.rule2 = ctrl.Rule(error ['ernp'], cali ['med'])
    self.rule3 = ctrl.Rule(error ['erz'], cali ['med'])
    self.rule4 = ctrl.Rule(error ['erpp'], cali ['abi'])
    self.rule5 = ctrl.Rule(error ['erpg'], cali ['abi'])
    self.rule6 = ctrl.Rule(error ['erng'], fria ['abi'])
    self.rule7 = ctrl.Rule(error ['ernp'], fria ['abi'])
    self.rule8 = ctrl.Rule(error ['erz'], fria ['med'])
    self.rule9 = ctrl.Rule(error ['erpp'], fria ['med'])
    self.rule10 = ctrl.Rule(error ['erpg'], fria ['cerr'])

```

Figura 55 Código de control difuso parte 3 en Visual Studio en lenguaje Python.

```

def calcular_salida(self, er):
    self.cali_ctrl.input['error'] = er
    self.fria_ctrl.input['error'] = er

    # Cálculo difuso
    self.cali_ctrl.compute()
    self.fria_ctrl.compute()

    # Devolver resultados
    resultado_cali = self.cali_ctrl.output['cali']
    resultado_fria = self.fria_ctrl.output['fria']
    return resultado_cali, resultado_fria

class UserInput:
    @staticmethod
    def obtener_valor_deseado():
        return float(input("Ingresa el valor deseado: "))

class FuzzySystem:
    def __init__(self):
        self.fuzzy_logic_controller = FuzzyLogicController()
        self.user_input = UserInput()

```

Figura 56 Código de control difuso parte 4 en Visual Studio en lenguaje Python.

```

fuzzy_system = FuzzySystem()

try:
    er=0
    # Obtener valor deseado del usuario
    valor_deseado = FuzzySystem().user_input.obtener_valor_deseado()
    while True:
        valor_sensado = float(input("Ingresa el valor sensado: "))
        er=(valor_deseado - valor_sensado)

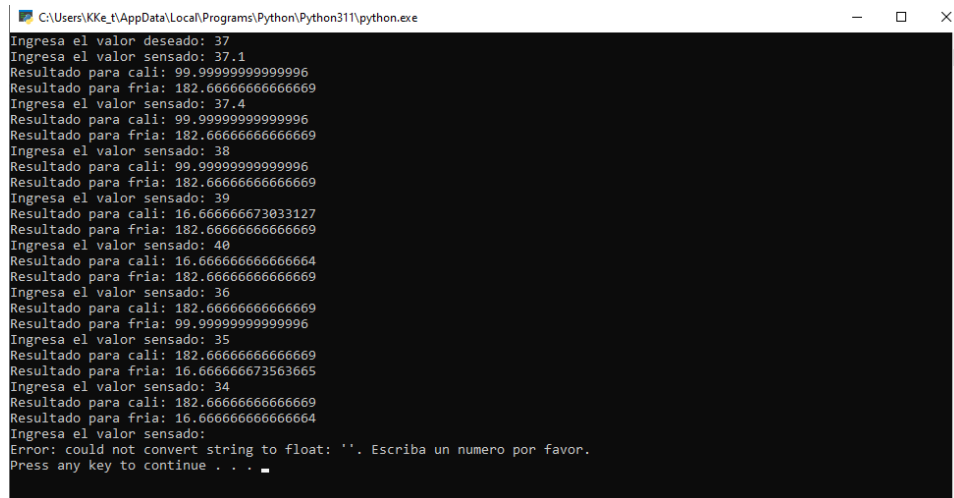
        # Calcular resultados utilizando la lógica difusa
        resultado_cali, resultado_fria = FuzzySystem().fuzzy_logic_controller.calcular_salida(er)
    # Resultados
    print(f"Resultado para cali: {resultado_cali}")
    print(f"Resultado para fria: {resultado_fria}")

except UnicodeDecodeError:
    print("Error de decodificación. Verifica la codificación del archivo.")
except ValueError as ve:
    print(f"Error: {ve}. Escriba un número por favor.")

```

Figura 57 Código de control difuso parte 4 en Visual Studio en lenguaje Python.

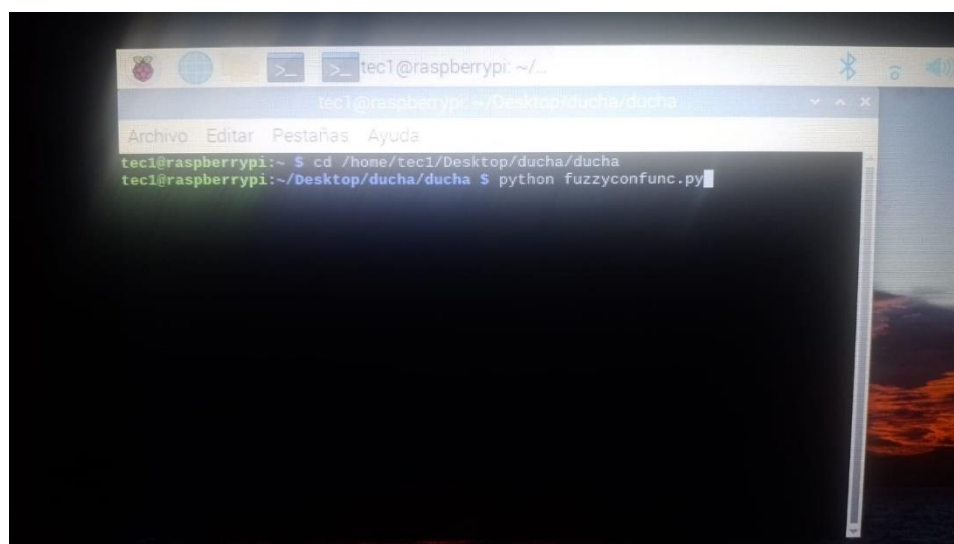
En la figura 58 se muestra la simulación del control difuso dentro de la laptop.



```
C:\Users\KKe_t\AppData\Local\Programs\Python\Python311\python.exe
Ingresar el valor deseado: 37
Ingresar el valor sensado: 37.1
Resultado para cali: 99.999999999999996
Resultado para fria: 182.66666666666669
Ingresar el valor sensado: 37.4
Resultado para cali: 99.999999999999996
Resultado para fria: 182.66666666666669
Ingresar el valor sensado: 38
Resultado para cali: 99.999999999999996
Resultado para fria: 182.66666666666669
Ingresar el valor sensado: 39
Resultado para cali: 16.6666666673033127
Resultado para fria: 182.66666666666669
Ingresar el valor sensado: 40
Resultado para cali: 16.666666666666664
Resultado para fria: 182.66666666666669
Ingresar el valor sensado: 48
Resultado para cali: 182.66666666666669
Resultado para fria: 99.999999999999996
Ingresar el valor sensado: 35
Resultado para cali: 182.66666666666669
Resultado para fria: 16.6666666673563665
Ingresar el valor sensado: 34
Resultado para cali: 182.66666666666669
Resultado para fria: 16.666666666666664
Ingresar el valor sensado:
Error: could not convert string to float: \'. Escriba un numero por favor.
Press any key to continue . . .
```

*Figura 58 Simulación de control difuso en la terminal de la laptop, desde Visual Studio.*

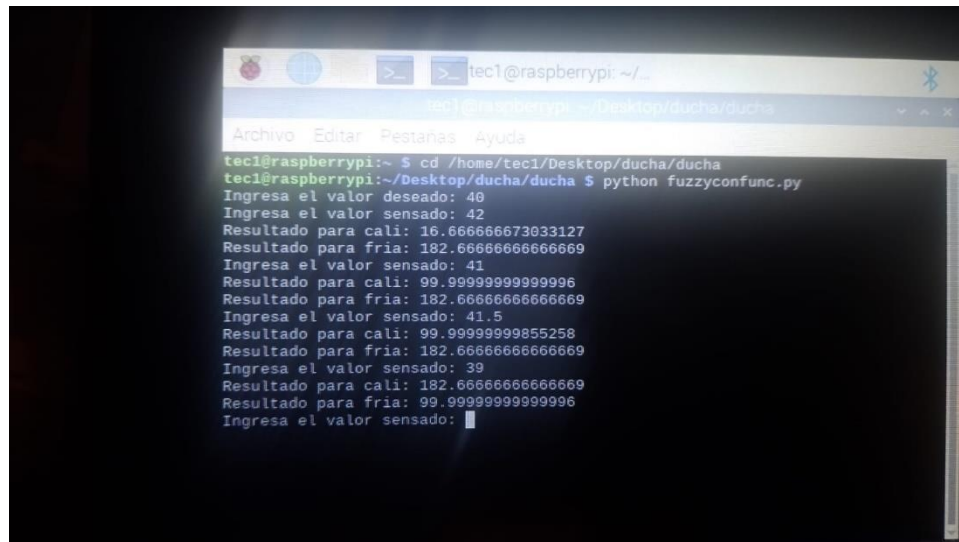
Para utilizar dicho código en la Raspberry pi, es necesario ejecutarlo desde la terminal, mediante los comandos de cambio de directorio a la carpeta donde se encuentra el archivo con extensión .py “cd /home/tec1/Desktop/ducha/ducha” y el de ejecución “python fuzzyfunc.py”, como se muestra en la figura 59.



```
tec1@raspberrypi: ~/Desktop/ducha/ducha
Archivo Editar Pestañas Ayuda
tec1@raspberrypi:~ $ cd /home/tec1/Desktop/ducha/ducha
tec1@raspberrypi:~/Desktop/ducha/ducha $ python fuzzyfunc.py
```

*Figura 59 Ejecución desde la terminal de la Raspberry pi.*

A continuación, en la figura 60 se muestra la simulación del control difuso dentro de la Raspberry pi.



```
tec1@raspberrypi: ~/...
tec1@raspberrypi: ~/Desktop/ducha/ducha
Archivo Editar Pestañas Ayuda
tec1@raspberrypi:~$ cd /home/tec1/Desktop/ducha/ducha
tec1@raspberrypi:~/Desktop/ducha/ducha$ python fuzzyconfunc.py
Ingresa el valor deseado: 40
Ingresa el valor sensado: 42
Resultado para cali: 16.666666673033127
Resultado para fria: 182.66666666666669
Ingresa el valor sensado: 41
Resultado para cali: 99.99999999999996
Resultado para fria: 182.66666666666669
Ingresa el valor sensado: 41.5
Resultado para cali: 99.9999999855258
Resultado para fria: 182.66666666666669
Ingresa el valor sensado: 39
Resultado para cali: 182.66666666666669
Resultado para fria: 99.99999999999996
Ingresa el valor sensado: █
```

*Figura 60 Simulación realizada en Raspberry pi.*

Como se puede observar, los valores son similares a la simulación dentro de la laptop.

## 3.2 Diseño de circuito electrónico

Para el diseño del circuito electrónico, utilizamos guías de diversos foros y las hojas de datos de cada componente del sistema, considerando las conexiones y alimentación que requieren para su funcionamiento y su correcto envío de datos.

Se consideraron los requisitos que requiere el proyecto, como la capacidad de mover las válvulas de CCPVC, la impermeabilidad de los componentes para evitar cortos circuitos, la compatibilidad de los componentes con la Raspberry pi y que su instalación resulte relativamente sencilla para el entorno en el que serán implementados.

Dentro de los componentes que se seleccionaron para integrar al sistema se encuentran sensores de temperatura DS18B20, sensores de flujo Yf-s201, Motores a pasos NEMA 17 con sus respectivos drivers TB6600, la propia Raspberry pi 4 con su pantalla táctil preinstalada.

A continuación, en la figura 61 se muestra el diagrama representativo de las conexiones de cada uno de los componentes del sistema con la Raspberry pi

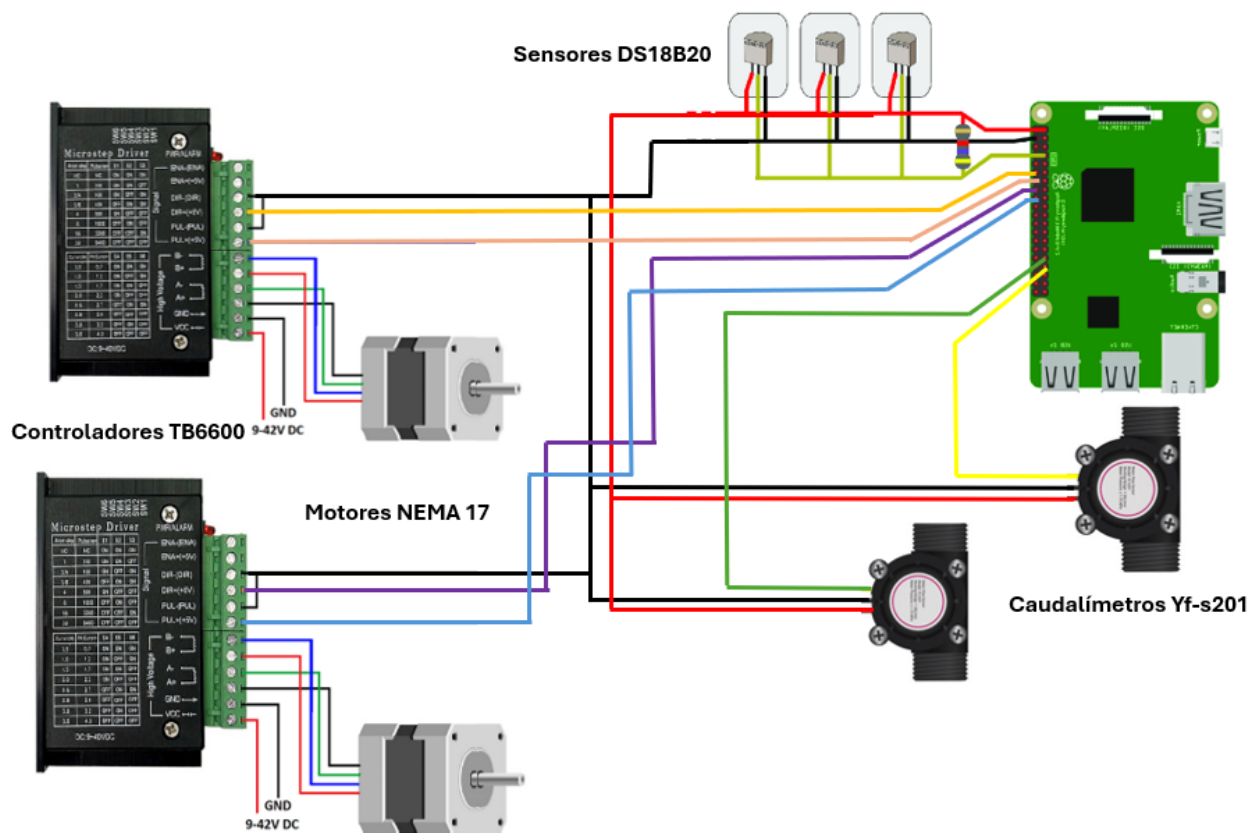


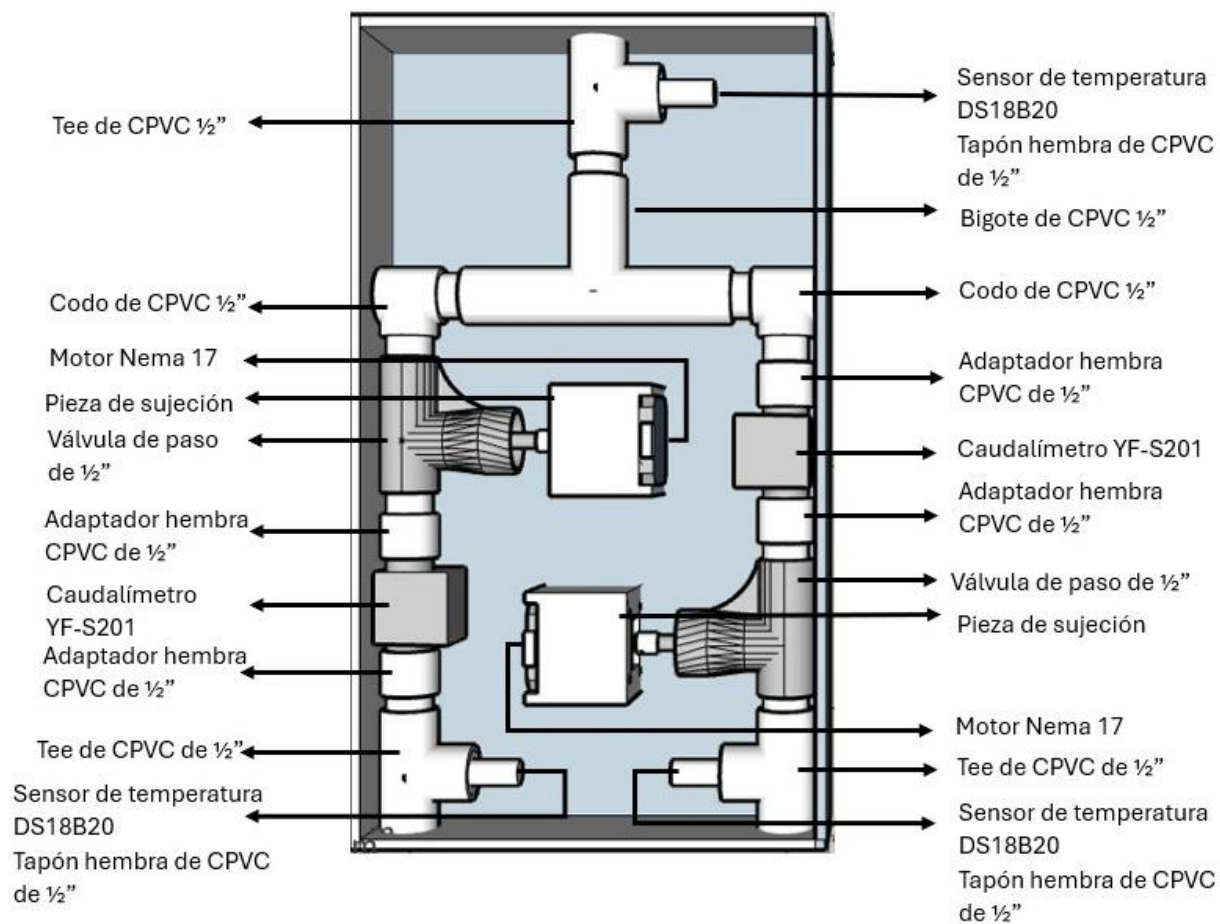
Figura 61 Diagrama del circuito electrónico conectado a la Raspberry pi.

### 3.3 Creación en 3D del prototipo por medio de software.

Para la creación 3D del prototipo utilizamos el programa SketchUp, un software que puede descargarse o utilizarse en línea, pero con menos herramientas, sin embargo, para la creación de este prototipo no son necesarias herramientas muy complejas. Se deben conocer las medidas reales de los componentes a utilizar para utilizarlas al momento del ensamblaje físico.

Se utilizan figuras sencillas como cilindros y prismas para representar figuras con curvas complejas, sin embargo, se consideran medidas reales para tener un margen de error mínimo y poder adaptar las piezas representadas según corresponda. En la figura 62 se muestran las piezas que componen al esquema 3D y en las figuras 63 a la 65 se pueden observar diversas perspectivas del modelo con sus medidas.





*Figura 62 Vista frontal de prototipo en 3D realizado con medidas reales.*

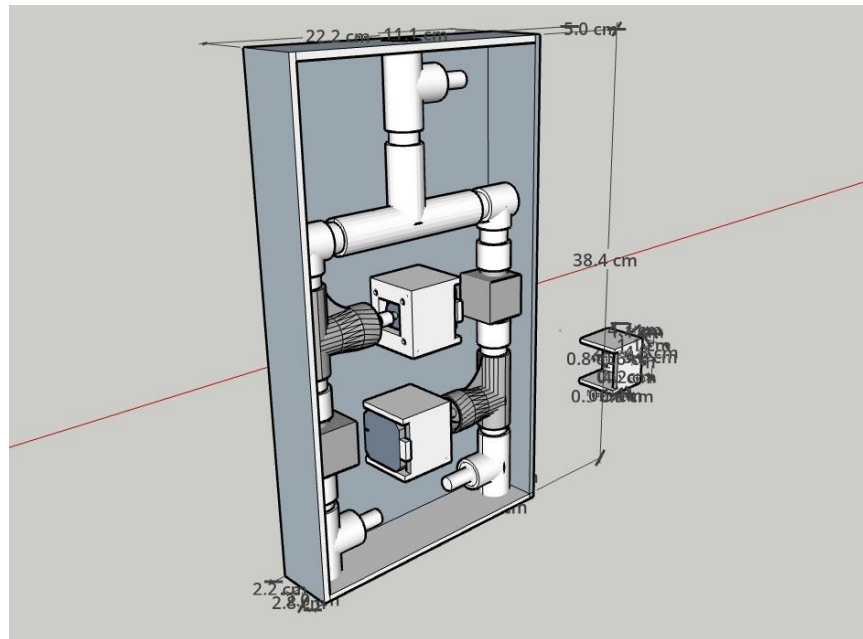


Figura 63 Vista lateral de prototipo en 3D con medidas reales.

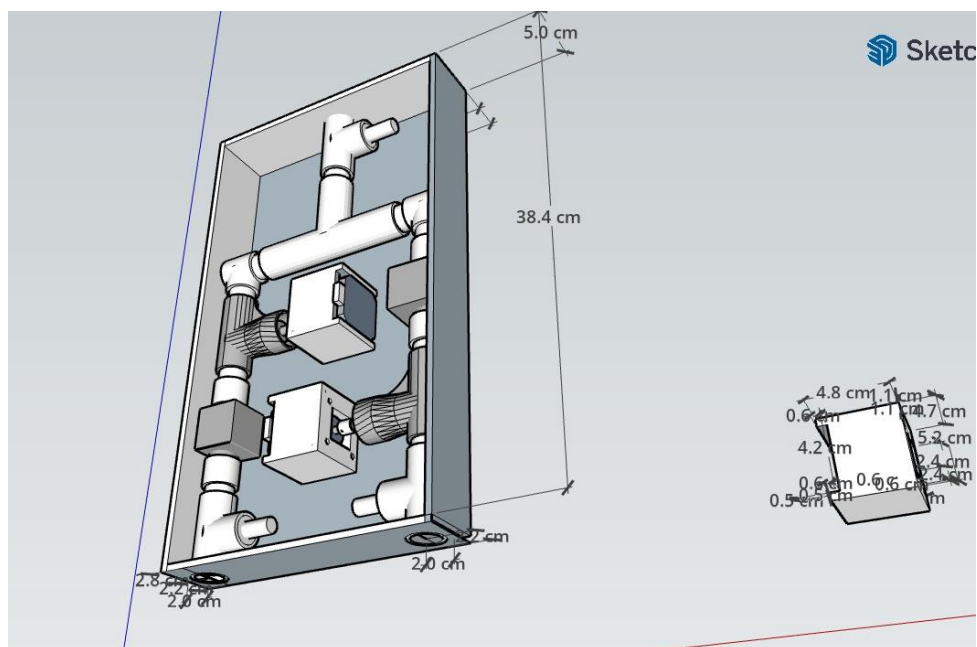
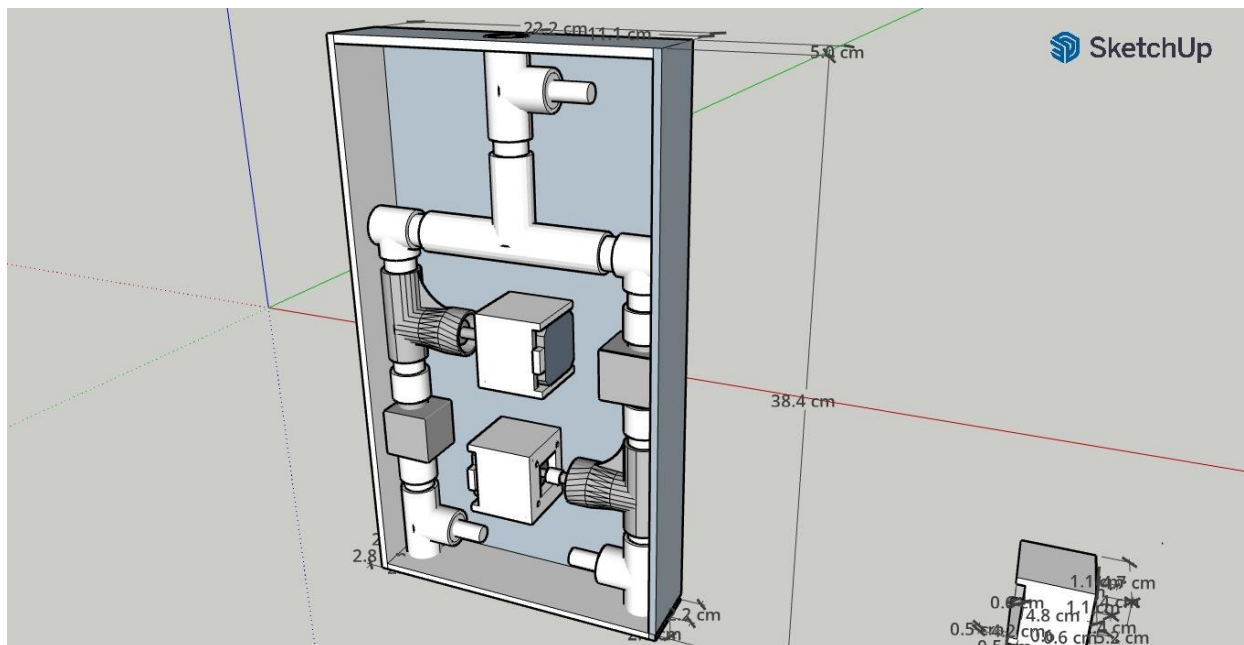


Figura 64 Vista inferior de prototipo en 3D con medidas reales.



*Figura 65 Vista superior de prototipo en 3D con medidas reales.*

### **3.4 Programación de interfaz de usuario en el panel táctil y aplicación web.**

La interfaz de usuario se realiza por medio de la programación de Python a través de la RaspberPi, haciendo uso de la librería Tkinter. Se desarrolla el código primero dentro de la computadora personal, para tener mayor comodidad y al no haber necesidad de utilizar pines GPIO se puede ejecutar sin problemas en ella, sin embargo, las pruebas de funcionalidad táctil se realizan dentro de la Raspberry pi. A continuación, en las figuras 66 a la 75 se muestra es desarrollo del programa de la interfaz, con valores preestablecidos de las variables a utilizar en el lugar que ocuparán los sensores al introducirse en el código, puesto que solamente se desarrolla la interfaz.

```

1  import tkinter as tk
2  import datetime
3  import time
4  import re
5
6  a = 0.133 # caudal 1
7  b = 0.133 # caudal 2
8  iniciar_sistema = False
9  tempdes = ""
10 tiempomin = ""
11 tempsen = 0 # sensor temp mez
12 cali = 0
13 fria = 0
14 minut = 0
15 seg = 0
16 fecha = ""
17 hora = ""
18
19 class Reloj:
20     def __init__(self):
21         self.fecha = datetime.date.today()
22         ahora = datetime.datetime.now()
23         self.hora = ahora.strftime("%H:%M:%S")
24
25     def actualizar_hora(self):
26         self.fecha = datetime.date.today()
27         ahora = datetime.datetime.now()
28         self.hora = ahora.strftime("%H:%M:%S")

```

Figura 66 Código de interfaz táctil de usuario parte 1 en Visual Studio con Python

```

30     def get_fecha(self):
31         return self.fecha
32
33     def get_hora(self):
34         return self.hora
35
36 class NumericalKeyboard:
37     def __init__(self, master, instancia_reloj):
38         self.instancia_reloj = instancia_reloj
39         self.fecha = self.instancia_reloj.get_fecha()
40         self.hora = self.instancia_reloj.get_hora()
41
42         self.entry_var = tk.StringVar()
43         self.tempdes = tk.StringVar()
44         self.tiempomin = tk.StringVar()
45         self.tiemposeg = tk.StringVar()
46         self.fechita = tk.StringVar()
47         self.horita = tk.StringVar()
48         self.entry_var.set("")
49         self.tempdes.set("TEMP DESEADA: 0 °C")
50         self.tiempomin.set("00:00 min")
51         self.master = master
52         self.master.title("Selector de Temperatura")
53         self.entry = tk.Entry(master, textvariable=self.entry_var, font=('Arial', 16), justify='center')
54         self.minuto = tk.Label(master, textvariable=self.tiempomin, font=('Arial', 11), justify='right')
55         self.letrero = tk.Label(master, text="SELECCIONAR °C y TIEMPO", font=('Arial', 14), justify='center')
56         #Label fecha y hora
57         self.fecha_label = tk.Label(master, textvariable=self.fechita, font=('Arial', 12, 'italic'), justify='center')

```

Figura 67 Código de interfaz táctil de usuario parte 2 en Visual Studio con Python



```

58 self.hora_label = tk.Label(master, textvariable=self.horita, font=('Arial', 12, 'italic'), justify='center')
59 #label de caudalímetros
60 self.caud1 = tk.Label(master, text="CAUDAL CALIENTE:", font=('Arial', 12), justify='left')
61 self.caud2 = tk.Label(master, text="CAUDAL FRIA:", font=('Arial', 12), justify='left')
62 self.caud1val = tk.Label(master, text=str(a), font=('Arial', 12), justify='left')
63 self.caud2val = tk.Label(master, text=str(b), font=('Arial', 12), justify='left')
64 self.caud1tx = tk.Label(master, text="L/m", font=('Arial', 12), justify='left')
65 self.caud2tx = tk.Label(master, text="L/m", font=('Arial', 12), justify='left')
66 #temperaturas label
67 self.temp2 = tk.Label(master, text="TEMP SENSADA:", font=('Arial', 12), justify='right')
68 self.temp1VAL = tk.Label(master, textvariable=self.tempdes, font=('Arial', 11), justify='center')
69 self.temp2VAL = tk.Label(master, text=tempdes, font=('Arial', 12), justify='center')
70 self.temp2tx = tk.Label(master, text="°C", font=('Arial', 12), justify='center')
71 self.temp3 = tk.Label(master, text="TEMP CALIENTE:", font=('Arial', 12), justify='right')
72 self.temp4 = tk.Label(master, text="TEMP FRIA:", font=('Arial', 12), justify='right')
73 self.temp3VAL = tk.Label(master, text=cali, font=('Arial', 12), justify='center')
74 self.temp4VAL = tk.Label(master, text=fria, font=('Arial', 12), justify='center')
75 self.temp3tx = tk.Label(master, text="°C", font=('Arial', 12), justify='center')
76 self.temp4tx = tk.Label(master, text="°C", font=('Arial', 12), justify='center')
77 #modular label
78 self.modup = tk.Label(master, text="SUBIR °C", font=('Arial', 8), justify='center')
79 self.moddn = tk.Label(master, text="BAJAR °C", font=('Arial', 8), justify='center')
80 #posicion barra de entrada y °C
81 self.entry.grid(row=1, column=0, columnspan=3, padx=1, pady=5, sticky='ew')
82 self.letrero.grid(row=0, column=0, columnspan=3, padx=5, pady=5, sticky='ew')
83 self.minuto.grid(row=2, column=5, columnspan=2, padx=1, pady=5, sticky='ew')
84 #posicion caudales
85 self.caud1.grid(row=3, column=3, columnspan=2, padx=5, pady=5, sticky='ew')

```

Figura 68 Código de interfaz táctil de usuario parte 3 en Visual Studio con Python

```

86 self.caud2.grid(row=4, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
87 self.caud1val.grid(row=3, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
88 self.caud2val.grid(row=4, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
89 self.caud1tx.grid(row=3, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
90 self.caud2tx.grid(row=4, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
91 #posicion temperaturas
92 self.temp2.grid(row=5, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
93 self.temp1VAL.grid(row=2, column=3, columnspan=1, padx=5, pady=5, sticky='ew')
94 self.temp2VAL.grid(row=5, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
95 self.temp2tx.grid(row=5, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
96 self.temp3.grid(row=6, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
97 self.temp4.grid(row=7, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
98 self.temp3VAL.grid(row=6, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
99 self.temp4VAL.grid(row=7, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
100 self.temp3tx.grid(row=6, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
101 self.temp4tx.grid(row=7, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
102 #posicion modular temp
103 self.modup.grid(row=6, column=0, columnspan=1, padx=0, pady=1, sticky='ew')
104 self.moddn.grid(row=7, column=0, columnspan=1, padx=0, pady=1, sticky='ew')
105 #grid hora y fecha
106 self.fecha_label.grid(row=0, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
107 self.hora_label.grid(row=0, column=5, columnspan=2, padx=5, pady=5, sticky='ew')
108
109 buttons = [
110     ('7', 2, 0), ('8', 2, 1), ('9', 2, 2),
111     ('4', 3, 0), ('5', 3, 1), ('6', 3, 2),
112     ('1', 4, 0), ('2', 4, 1), ('3', 4, 2),
113     ('.', 5, 0), ('0', 5, 1), ('+', 5, 2),

```

Figura 69 Código de interfaz táctil de usuario parte 4 en Visual Studio con Python

```

114         ('▲', 6, 1), ('↑', 6, 2),
115         ('▼', 7, 1), ('↓', 7, 2)
116     ]
117
118     for (text, row, col) in buttons:
119         if text == '←':
120             button = tk.Button(master, text=text, font=('Arial', 15), command=self.clear_entry)
121         else:
122             button = tk.Button(master, text=text, font=('Arial', 15), command=lambda t=text: self.on_button_click(t))
123             button.grid(row=row, column=col, columnspan=1, padx=1, pady=1, sticky='nsew')
124
125     buttons1 = [
126         ('SET °C', 1, 3), ('SET TIME', 1, 5),
127     ]
128
129     for (text, row, col) in buttons1:
130         button = tk.Button(master, text=text, font=('Arial', 12, 'bold'), command=lambda t=text: self.on_button_click(t))
131         button.grid(row=row, column=col, rowspan=1, columnspan=2, padx=1, pady=1, sticky='nsew')
132
133     buttons2 = [
134         ('INICIAR', 8, 5), ('CANCELAR', 8, 3),
135     ]
136
137     for (text, row, col) in buttons2:
138         button = tk.Button(master, text=text, font=('Arial Black', 14), command=lambda t=text: self.on_button_click(t))
139         button.grid(row=row, column=col, columnspan=2, padx=1, pady=1, sticky='nsew')
140         self.update_clock()
141     #Funciones de fecha, hora y temporizador. estan sincronizados para evitar error

```

Figura 70 Código de interfaz táctil de usuario parte 5 en Visual Studio con Python

```

142     def update_clock(self):
143         self.instancia_reloj.actualizar_hora()
144         fecha_valor = self.instancia_reloj.get_fecha()
145         hora_valor = self.instancia_reloj.get_hora()
146         self.fechita.set("Fecha: " + str(fecha_valor))
147         self.horita.set("Hora: " + str(hora_valor))
148         tiempo_restante = self.tiempomin.get()
149         numero_tiempo_restante = re.findall(r'\d+\.\d+|\d+', tiempo_restante)
150         minutos_rest=int(numero_tiempo_restante[0])
151         segundos_rest=int(numero_tiempo_restante[1])
152         tiempo_rest_en_seg=(minutos_rest*60)+segundos_rest
153         #print (tiempo_rest_en_seg)
154         if tiempo_rest_en_seg > 0:
155             if iniciar_sistema==True:
156                 tiempo_rest_en_seg -=1
157             else:
158                 iniciar_sistema=False
159             tiempo_decimal=tiempo_rest_en_seg/60
160             minuterio = int(tiempo_decimal)
161             segundero = int(round(((tiempo_decimal - minuterio)*60),1))
162             if segundero < 10:
163                 self.tiempomin.set(str(minuterio) + ":0" + str(segundero) + " min")
164             else:
165                 self.tiempomin.set(str(minuterio)+":"+str(segundero)+ " min")
166             self.master.after(1000, self.update_clock)
167     def on_button_click(self, value):
168         global iniciar_sistema, minut, seg, tempdes
169         if value == 'SET °C':

```

Figura 71 Código de interfaz táctil de usuario parte 6 en Visual Studio con Python



```

170     try:
171         result = eval(self.entry_var.get())
172         self.entry_var.set(str(result))
173         self.tempdes.set("TEMP DESEADA: "+ str(result)+" °C")
174         print (result)
175         self.clear_entry()
176     except:
177         self.entry_var.set("Error, ingresa numero")
178 elif value == 'SET TIME':
179     try:
180         result = eval(self.entry_var.get())
181         minut = int(result)
182         segun = round((result - minut), 3)
183         seg = int(segun * 60)
184         if seg < 10:
185             self.tiempomin.set(str(minut) + ":0" + str(seg) + " min")
186         else:
187             self.tiempomin.set(str(minut) + ":" + str(seg) + " min")
188         print (minut, ":", seg, " min")
189         self.clear_entry()
190         if minut == 0 and seg == 0:
191             iniciar_sistema = False
192             print ("Sistema: ", iniciar_sistema)
193     except:
194         self.entry_var.set("Error, ingresa numero")
195 elif value == 'INICIAR':
196     iniciar_sistema = True
197     print ("Sistema: ", iniciar_sistema)

```

Figura 72 Código de interfaz táctil de usuario parte 7 en Visual Studio con Python

```

198 elif value == 'CANCELAR':
199     minut = 0
200     seg = 0
201     result = 0
202     self.tiempomin.set("0" + str(minut) + ":0" + str(seg) + " min")
203     self.tempdes.set("TEMP DESEADA: "+ str(result)+" °C")
204     self.clear_entry()
205     iniciar_sistema = False
206     print ("Sistema: ", iniciar_sistema)
207 elif value == '←':
208     self.clear_entry()
209 elif value == '↓':
210     self.mdon = self.tempdes.get()
211     numdn = re.findall(r'\d+\.\d+|\d+', self.mdon)
212     numa = ''.join(numdn)
213     self.mdon_val = float(numa)
214     self.mdon_val = self.mdon_val - 0.1
215     don_red = round(self.mdon_val, 3)
216     if don_red <= 0:
217         don_red = 0
218     print (don_red)
219     self.tempdes.set("TEMP DESEADA: "+ str(don_red)+" °C")
220     value = ' '
221     self.clear_entry()
222 elif value == '↑':
223     self.mup = self.tempdes.get()
224     numup = re.findall(r'\d+\.\d+|\d+', self.mup)
225     numb = ''.join(numup)

```

Figura 73 Código de interfaz táctil de usuario parte 8 en Visual Studio con Python

```

226         self.mup_val = float(numb)
227         self.mup_val = self.mup_val + 0.1
228         up_red = round(self.mup_val, 3)
229         print (up_red)
230         self.tempdes.set("TEMP DESEADA: " + str(up_red)+" °C")
231         value = ' '
232         self.clear_entry()
233     elif value == '▼':
234         self.mdon = self.tempdes.get()
235         numdn = re.findall(r'\d+\.\d+|\d+', self.mdon)
236         numa = ''.join(numdn)
237         self.mdon_val = float(numa)
238         self.mdon_val -= 1
239         if self.mdon_val < 0:
240             self.mdon_val = 0
241         print (self.mdon_val)
242         don_red = round(self.mdon_val, 3)
243         self.tempdes.set("TEMP DESEADA: " + str(don_red)+" °C")
244         value = ' '
245         self.clear_entry()
246     elif value == '▲':
247         self.mup = self.tempdes.get()
248         numup = re.findall(r'\d+\.\d+|\d+', self.mup)
249         numb = ''.join(numup)
250         self.mup_val = float(numb)
251         self.mup_val += 1
252         print (self.mup_val)
253         up_red = round(self.mup_val, 3)

```

Figura 74 Código de interfaz táctil de usuario parte 9 en Visual Studio con Python

```

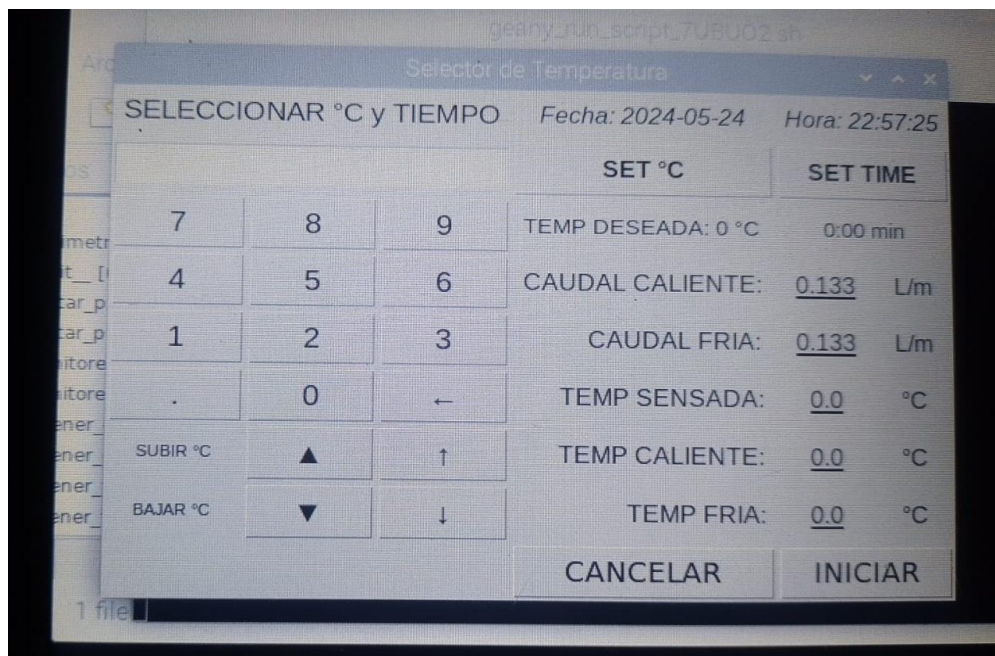
254         self.tempdes.set("TEMP DESEADA: " + str(up_red)+" °C")
255         value = ' '
256         self.clear_entry()
257     else:
258         current_text = self.entry_var.get()
259         if value == '.':
260             if '.' not in current_text:
261                 self.entry_var.set(current_text + value)
262             else:
263                 self.entry_var.set(current_text + value)
264
265     def clear_entry(self):
266         self.entry_var.set("")
267
268     instancia_reloj=Reloj()
269
270     def main():
271         root = tk.Tk()
272         app = NumericalKeyboard(root, instancia_reloj)
273         root.mainloop()
274
275     if __name__ == "__main__":
276         main()

```

Figura 75 Código de interfaz táctil de usuario parte 10 en Visual Studio con Python



En esta interfaz de usuario se consideraron elementos sencillos de comprender, como un teclado donde poder introducir el valor numérico de la temperatura y el tiempo del temporizador, además un par de botones para aumentar o disminuir la temperatura en valores de unidad o en valores de decimales. También tiene un apartado de fecha y hora para que, al ser utilizado, pueda conocerse el momento en el que empieza y termina el baño del usuario. Además, tiene una sección de monitorización de los caudales de agua caliente y fría y de los tres sensores de temperatura de agua (caliente, fría y mezcla). Esta interfaz puede visualizarse en la figura 76.



*Figura 76 Interfaz de usuario.*

Para el desarrollo de la aplicación web se optó por un diseño aún más simple, pues en los dispositivos móviles es posible acceder a un teclado y visualizar la hora. Haciendo uso de la programación en HTML y su implementación dentro del código en Python (usando la librería Flask) para que modifique las mismas variables que la interfaz creada mediante Tkinter. En las figuras 77 a la 80.

```

1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta name="viewport" content="width=device-width, initial-scale=1.0">
6    <title>Selector de Temperatura</title>
7    <style>
8      body {
9        font-family: Arial, sans-serif;
10      }
11
12      .container {
13        width: 70%;
14        margin: auto;
15      }
16
17      .row {
18        display: flex;
19        justify-content: space-between;
20        margin-bottom: 10px;
21      }
22
23      .column {
24        flex: 1;
25        text-align: center;
26      }
27
28      .column input, .column button {
29        width: 100%;

```

Figura 77 Código de interfaz web de usuario parte 1 en Visual Studio con HTML

```

30      padding: 10px;
31      margin: 5px 0;
32    }
33
34    .keyboard {
35      display: flex;
36      flex-wrap: wrap;
37      justify-content: center;
38      margin-top: 10px;
39    }
40
41    .key {
42      width: 30%;
43      padding: 8px;
44      margin: 2px;
45      background-color: #f0f0f0;
46      border: 1px solid #ccc;
47      cursor: pointer;
48      text-align: center;
49      font-size: 16px;
50      user-select: none;
51    }
52  </style>
53  <script>
54    function insertNumber(num) {
55      var inputField = document.getElementById("temperature");
56      inputField.value += num;
57    }
58  </script>

```

Figura 78 Código de interfaz web de usuario parte 2 en Visual Studio con HTML

```

59     function clearInput() {
60         var inputField = document.getElementById("temperature");
61         inputField.value = "";
62     }
63     </script>
64 </head>
65 <body>
66     <div class="container">
67         <h1>Selector de Temperatura </h1>
68         <h2>Fecha: {{ fecha }} | Hora: {{ hora }} </h2>
69         <form action="{{ url_for('set_temperature') }}" method="post">
70             <div class="row">
71                 <div class="column">
72                     <label for="temperature">Temperatura Deseada:</label>
73                     <input type="text" id="temperature" name="temperature" value="{{ temperatura_sensada }}">
74                 </div>
75                 <div class="column">
76                     <label for="time">Tiempo (min:seg):</label>
77                     <input type="text" id="time" name="time" value="{{ minut }}:{{ seg }}">
78                 </div>
79             </div>
80             <div class="row">
81                 <div class="column">
82                     <button type="submit" formaction="{{ url_for('start') }}">Iniciar</button>
83                 </div>
84                 <div class="column">
85                     <button type="submit" formaction="{{ url_for('cancel') }}">Cancelar</button>
86                 </div>
87             </div>

```

Figura 79 Código de interfaz web de usuario parte 3 en Visual Studio con HTML

```

88     </form>
89     <h2>Estado Actual</h2>
90     <p>Temperatura Sensada: {{ temperatura_sensada }} °C</p>
91     <p>Temperatura Caliente: {{ temperatura_caliente }} °C</p>
92     <p>Temperatura Fría: {{ temperatura_fria }} °C</p>
93     <p>Caudal 1: {{ caudal1 }} L/m</p>
94     <p>Caudal 2: {{ caudal2 }} L/m</p>
95 </div>
96 </body>
97 </html>

```

Figura 80 Código de interfaz web de usuario parte 3 en Visual Studio con HTML

La aplicación web de la figura 81 consta de un apartado donde se muestra la fecha y hora, también se muestra la sección de escritura de la temperatura deseada y del temporizador, además de una zona de monitorización de los mismos sensores que se muestran en la interfaz táctil de la Raspberry pi



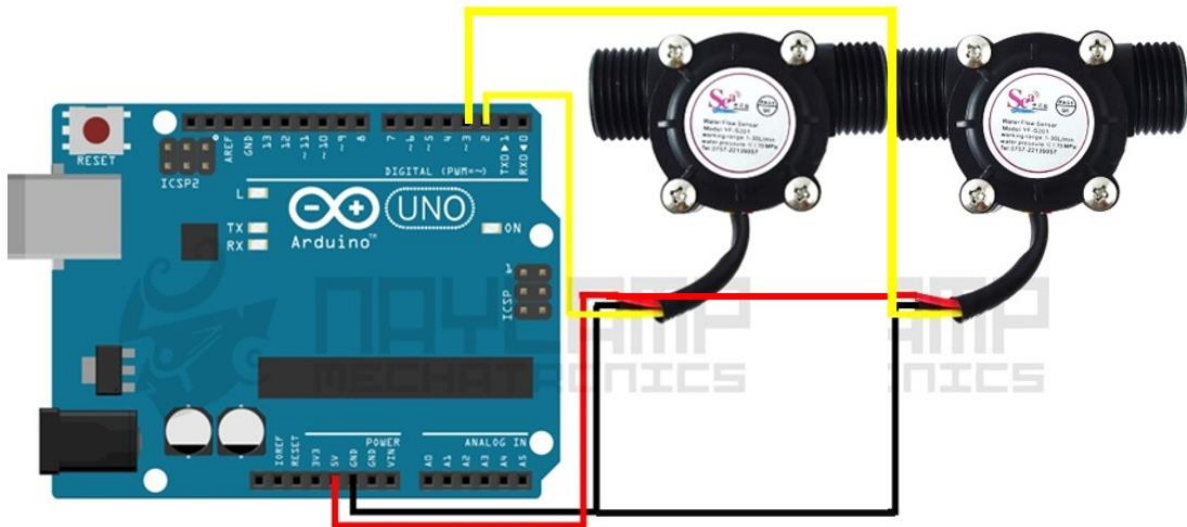
*Figura 81 Aplicación web.*

## 3.5 Pruebas preliminares

### 3.5.1 Prueba unitaria de integración para caudalímetro.

Para realizar la prueba unitaria de integración del caudalímetro, se consideró realizar pruebas en Arduino antes que en la Raspberry pi, para poder comparar los datos de las mediciones y realizar ajustes en el código en caso de haber incongruencias con los datos

En la figura 82 se muestra el diagrama esquemático que utilizamos en esta parte de la prueba implementada con Arduino.



*Figura 82 Diagrama esquemático de Arduino UNO y caudalímetros.*

Para verificar el funcionamiento de los caudalímetros se necesitó una superficie alta como la que se muestra en la figura 83 y 84, pues se realizó el vaciado de un par de galones que representarían las fuentes de agua caliente y fría en pruebas siguientes.



*Figura 83 Superficie alta para medir caudal.*



*Figura 84 Conexiones físicas de Arduino y caudalímetros.*

Para el desarrollo en el entorno de Arduino, no es necesario tener instaladas las librerías especiales. Por lo cual, podemos implementar el código directamente tal como muestra en las figuras 85 a 87.

```

1  volatile int NumPulsos1; // variable para pulsos del primer caudalímetro
2  volatile int NumPulsos2; // variable para pulsos del segundo caudalímetro
3  int PinSensor1 = 2;
4  int PinSensor2 = 3;
5  float factor_conversion = 7.5;
6  // Función que se ejecuta en interrupción
7  void ContarPulsos1() {
8      NumPulsos1++; // incrementamos pulsos del primer caudalímetro
9  }
10
11 void ContarPulsos2() {
12     NumPulsos2++; // incrementamos pulsos del segundo caudalímetro
13 }

```

*Figura 85 Parte 1 de código en IDE de Arduino UNO.*



```

15 // --- Función para obtener frecuencia de los pulsos ---
16 int ObtenerFrecuencia(int pin) {
17     int frecuencia;
18     if (pin == PinSensor1) {
19         NumPulsos1 = 0; // reiniciamos el número de pulsos del primer caudalímetro
20     } else if (pin == PinSensor2) {
21         NumPulsos2 = 0; // reiniciamos el número de pulsos del segundo caudalímetro
22     }
23     interrupts(); // Habilitamos las interrupciones
24     delay(1000);
25     noInterrupts(); // Deshabilitamos las interrupciones
26     if (pin == PinSensor1) {
27         frecuencia = NumPulsos1; // Hz del primer caudalímetro
28     } else if (pin == PinSensor2) {
29         frecuencia = NumPulsos2; // Hz del segundo caudalímetro
30     }
31     return frecuencia;
32 }

```

*Figura 86 Parte 2 de código en IDE de Arduino UNO.*

```

34 void setup() {
35     Serial.begin(9600);
36     pinMode(PinSensor1, INPUT);
37     pinMode(PinSensor2, INPUT);
38     attachInterrupt(digitalPinToInterrupt(PinSensor1), ContarPulsos1, RISING); // Interrupción para el primer caudalímetro
39     attachInterrupt(digitalPinToInterrupt(PinSensor2), ContarPulsos2, RISING); // Interrupción para el segundo caudalímetro
40 }
41
42 void loop() {
43     float frecuencia1 = ObtenerFrecuencia(PinSensor1); // obtenemos la Frecuencia del primer caudalímetro en Hz
44     float frecuencia2 = ObtenerFrecuencia(PinSensor2); // obtenemos la Frecuencia del segundo caudalímetro en Hz
45     float caudal_L_m1 = frecuencia1 / factor_conversion; // caudal en L/m del primer caudalímetro
46     float caudal_L_h1 = caudal_L_m1 * 60; // caudal en L/h del primer caudalímetro
47     float caudal_L_m2 = frecuencia2 / factor_conversion; // caudal en L/m del segundo caudalímetro
48     float caudal_L_h2 = caudal_L_m2 * 60; // caudal en L/h del segundo caudalímetro
49     Serial.print("Caudalímetro 1 -> FrecuenciaPulsos: ");
50     Serial.print(frecuencia1, 0);
51     Serial.print("Hz\tCaudal: ");
52     Serial.print(caudal_L_m1, 3);
53     Serial.print(" L/m\t");
54     Serial.print(caudal_L_h1, 3);
55     Serial.println(" L/h");
56     Serial.print("Caudalímetro 2 -> FrecuenciaPulsos: ");
57     Serial.print(frecuencia2, 0);
58     Serial.print("Hz\tCaudal: ");
59     Serial.print(caudal_L_m2, 3);
60     Serial.print(" L/m\t");
61     Serial.print(caudal_L_h2, 3);
62     Serial.println(" L/h");
63     delay(1000);
64 }

```

*Figura 87 Parte 3 de código en IDE de Arduino UNO.*

Una vez realizadas las conexiones correspondientes, al abrir el monitor serial se pueden observar las medidas obtenidas por los sensores de flujo YF-S201 como se muestra en la figura 88.

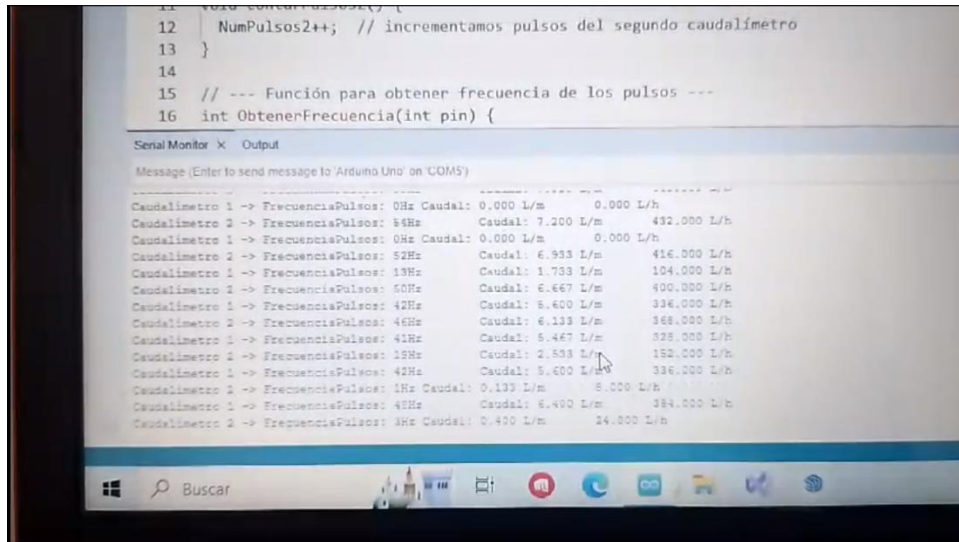


Figura 88 Valores obtenidos por los caudalímetros.

Para realizar la prueba preliminar en la Raspberry Pi, es necesario realizar el diagrama esquemático de cada uno de los elementos que se integraran en el circuito, por lo tanto, se procederá a conectar la Raspberry Pi como se muestra a continuación en la figura 89.

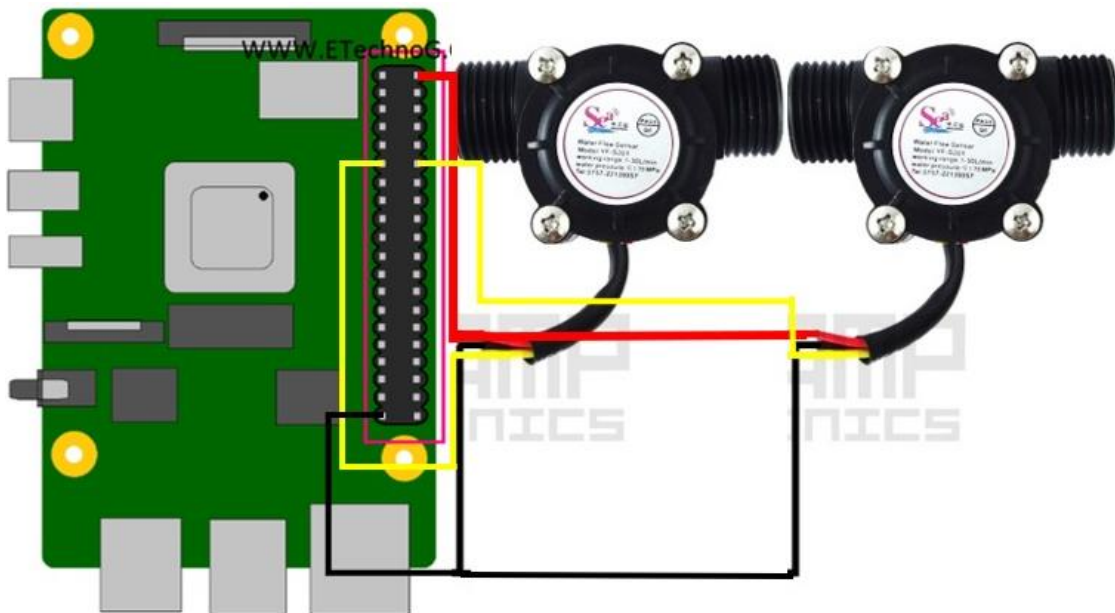


Figura 89 Diagrama esquemático de caudalímetros y Raspberry Pi.

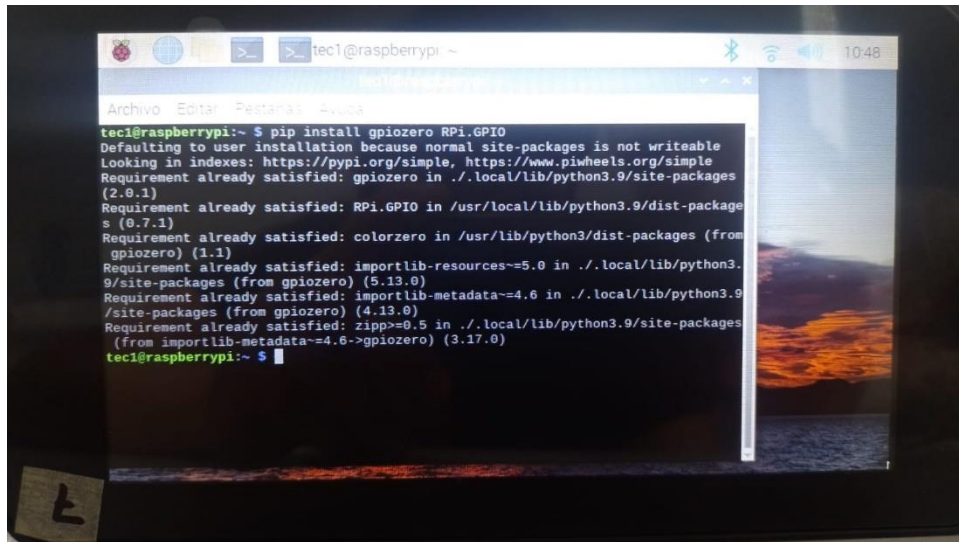


Realizamos las conexiones correspondientes a la Raspberry pi como se muestra en la figura 90, haciendo uso de tubos y acoplamientos roscados hembra de CCPVC y segmentos de manguera transparente que se une a los galones ya mencionados en la implementación con Arduino.



*Figura 90 Conexiones de Raspberry Pi y caudalímetros.*

El entorno de la Raspberry pi 4, es necesario tener instaladas las librerías necesarias, en este caso, **gpiozero**, **RPI.GPIO**. Para lo cual utilizamos el comando “pip install *nombre\_de\_la\_librería*” como se muestra a continuación en la figura 91.



*Figura 91 Instalación de librerías gpiozero,RPi GPIO.*

Una vez instaladas las librerías, podemos integrar el código en Visual Studio, pues como ya se ha mencionado, ofrece una mayor comodidad para programar que en la pantalla de la Raspberry pi, obteniendo el archivo con extensión “.py” que transferiremos a la Raspberry Pi como se muestra en las figuras 92 a 96.

```

import RPi.GPIO as GPIO
import threading
import time
import gpiozero

NumPulsos1 = 0
NumPulsos2 = 0
PinSensor1 = 17 # Primer caudalímetro
PinSensor2 = 18 # Segundo caudalímetro
factor_conversion = 7.5
frecuencia1 = 0
frecuencia2 = 0

lock1 = threading.Lock()
lock2 = threading.Lock()

def contar_pulsos1(channel):
    global NumPulsos1
    with lock1:
        NumPulsos1 += 1

def contar_pulsos2(channel):
    global NumPulsos2
    with lock2:
        NumPulsos2 += 1

```

*Figura 92 Parte 1 de código en Visual Studio en lenguaje Python.*

```
def obtener_frecuencial():
    global NumPulsos1
    with lock1:
        NumPulsos1 = 0
    GPIO.setup(PinSensor1, GPIO.OUT)
    GPIO.output(PinSensor1, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(PinSensor1, GPIO.LOW)
    GPIO.setup(PinSensor1, GPIO.IN)
    time.sleep(1)
    with lock1:
        frecuencial = NumPulsos1
    return NumPulsos1
```

Figura 93 Parte 2 de código en Visual Studio en lenguaje Python.

```
def obtener_frecuencia2():
    global NumPulsos2
    with lock2:
        NumPulsos2 = 0
    GPIO.setup(PinSensor2, GPIO.OUT)
    GPIO.output(PinSensor2, GPIO.HIGH)
    time.sleep(0.1)
    GPIO.output(PinSensor2, GPIO.LOW)
    GPIO.setup(PinSensor2, GPIO.IN)
    time.sleep(1)
    with lock2:
        frecuencia2 = NumPulsos2
    return NumPulsos2
```

Figura 94 Parte 3 de código en Visual Studio en lenguaje Python.

```
def monitorear_caudalimetro1():
    while True:
        frecuencial = obtener_frecuencial()
        caudal_L_m1 = frecuencial / factor_conversion
        caudal_L_h1 = caudal_L_m1 * 60
        print(f"Caudalímetro 1 - Frecuencia: {frecuencial} Hz\tCaudal: {caudal_L_m1:.3f} L/m\t{caudal_L_h1:.3f} L/h")
        time.sleep(1)

def monitorear_caudalimetro2():
    while True:
        frecuencia2 = obtener_frecuencia2()
        caudal_L_m2 = frecuencia2 / factor_conversion
        caudal_L_h2 = caudal_L_m2 * 60
        print(f"Caudalímetro 2 - Frecuencia: {frecuencia2} Hz\tCaudal: {caudal_L_m2:.3f} L/m\t{caudal_L_h2:.3f} L/h")
        time.sleep(1)
```

Figura 95 Parte 4 de código en Visual Studio en lenguaje Python.

```

# Configurar la biblioteca GPIO y los pines de los sensores
GPIO.setmode(GPIO.BCM)
GPIO.setup(PinSensor1, GPIO.OUT)
GPIO.setup(PinSensor2, GPIO.OUT)
GPIO.setup(PinSensor1, GPIO.IN)
GPIO.setup(PinSensor2, GPIO.IN)
GPIO.add_event_detect(PinSensor1, GPIO.RISING, callback=contar_pulsos1)
GPIO.add_event_detect(PinSensor2, GPIO.RISING, callback=contar_pulsos2)
# Crear hilos para monitorizar cada caudalimetro
hilo1 = threading.Thread(target=monitorear_caudalimetro1)
hilo2 = threading.Thread(target=monitorear_caudalimetro2)
# Iniciar los hilos
hilo1.start()
hilo2.start()

try:
    # Mantener el programa principal a la espera
    hilo1.join()
    hilo2.join()

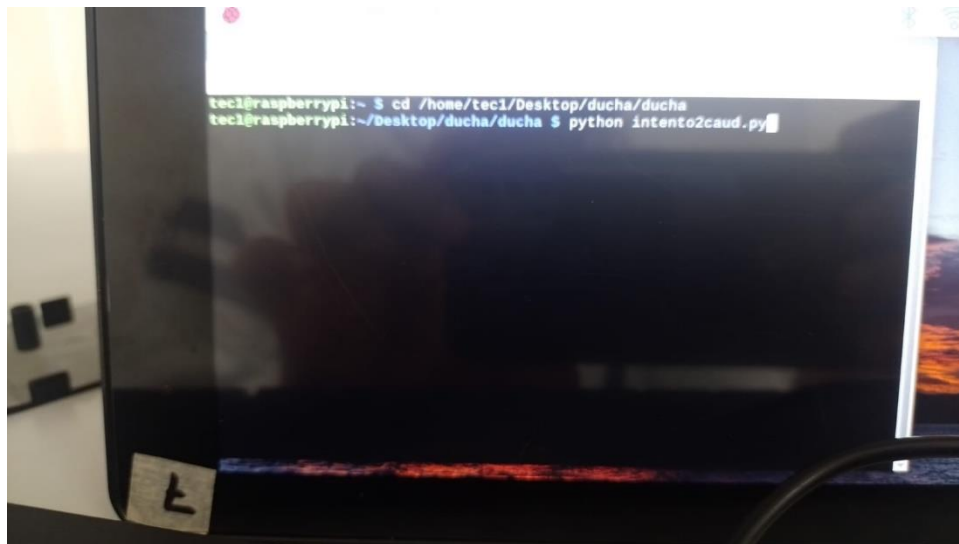
except KeyboardInterrupt:
    print("Programa detenido por el usuario") # Detenido al presionar cntrl+c

finally:
    GPIO.cleanup()

```

*Figura 96 Parte 5 de código en Visual Studio en lenguaje Python.*

Para utilizar dicho código en la Raspberry pi, es necesario ejecutarlo desde la terminal, mediante los comandos de cambio de directorio a la carpeta donde se encuentra el archivo con extensión .py “**cd /home/tec1/Desktop/ducha/ducha**” y el de ejecución “**python caudhilos.py**” como se muestra en la figura 97 y 98



*Figura 97 Comando de cambio de directorio.*



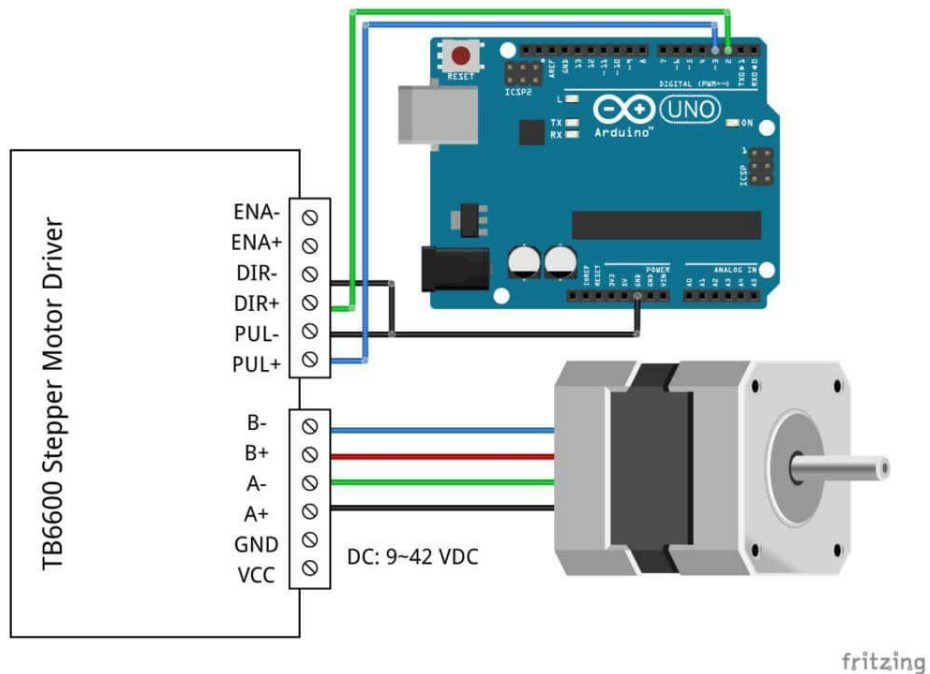
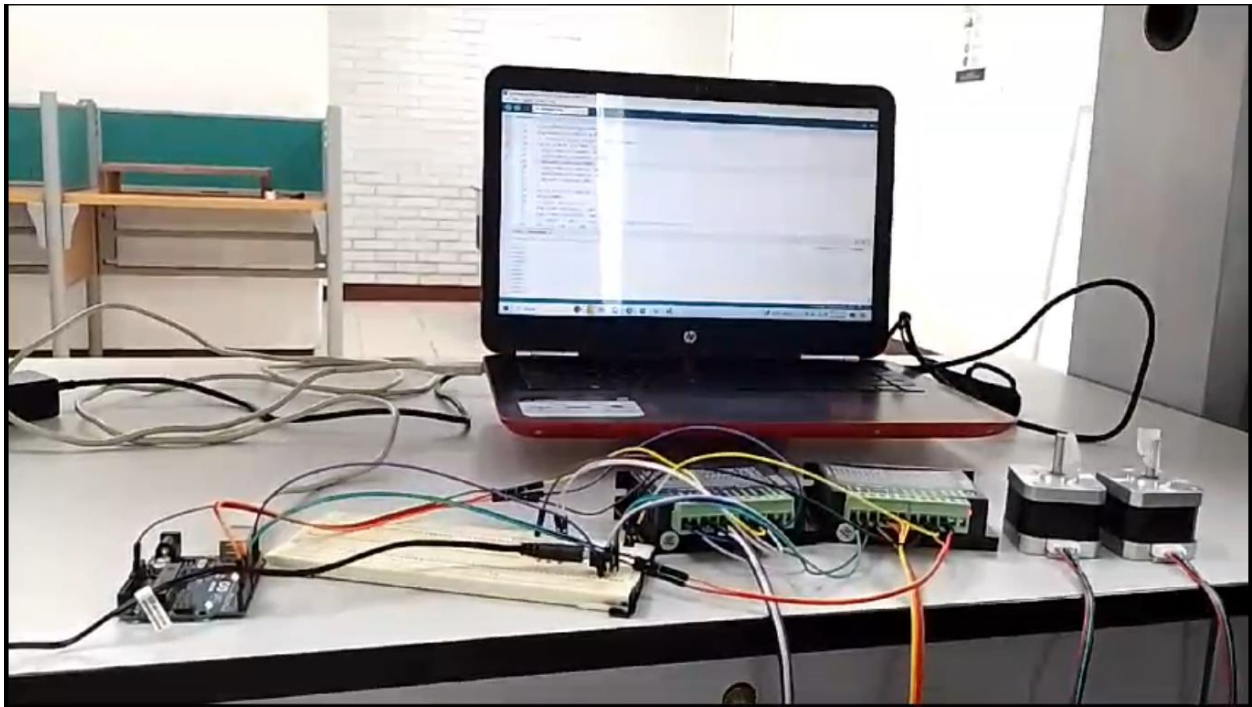


Figura 99 Diagrama Esquemático de Arduino UNO y motores nema 17.

Una vez identificados los pines de conexión, realizamos las conexiones necesarias tal como se muestra en la figura 100, revisando detalladamente que la configuración del controlador TB6600 sea la adecuada para el motor, el cual consume 0.5 A y tiene 200 pasos por revolución, por lo cual, la configuración para los DIP switch de ambos motores, según el fabricante, es: sw1=ON, sw2=ON, sw3=OFF, sw4= ON, sw5= ON, sw6= ON.

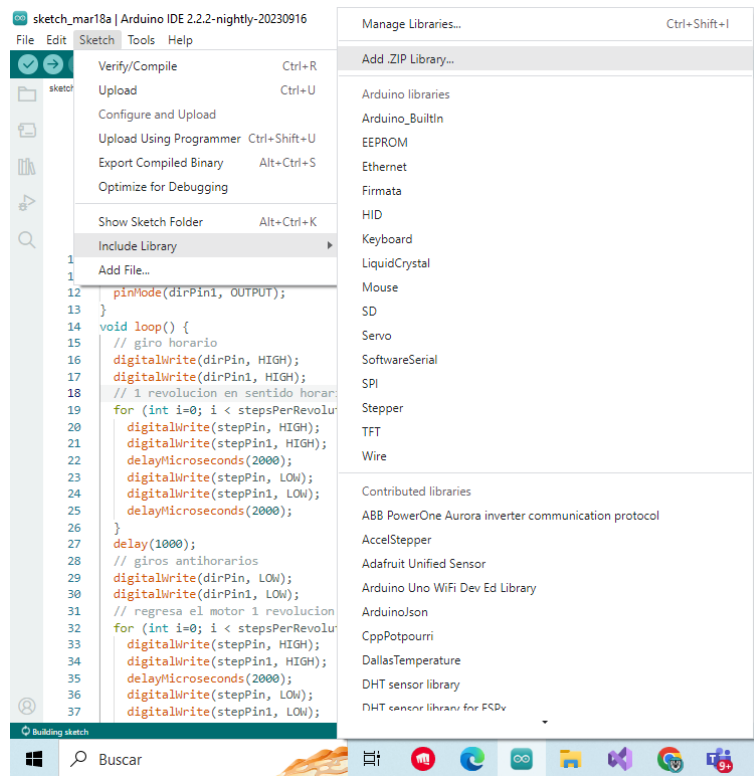


*Figura 100 Conexiones físicos de Arduino UNO y motores nema 17.*

Para poder utilizar el programa de prueba en Arduino, es necesario instalar la librería AccelStepper, para lo cual accedemos al siguiente enlace: <https://www.makerguides.com/wp-content/uploads/2019/02/AccelStepper-1.59.zip>.

Una vez descargado el archivo, lo incluimos a las librerías de Arduino como se muestra a continuación en la figura 101, dando clic en el apartado de “Sketch>Include Library>Add .ZIP Library”

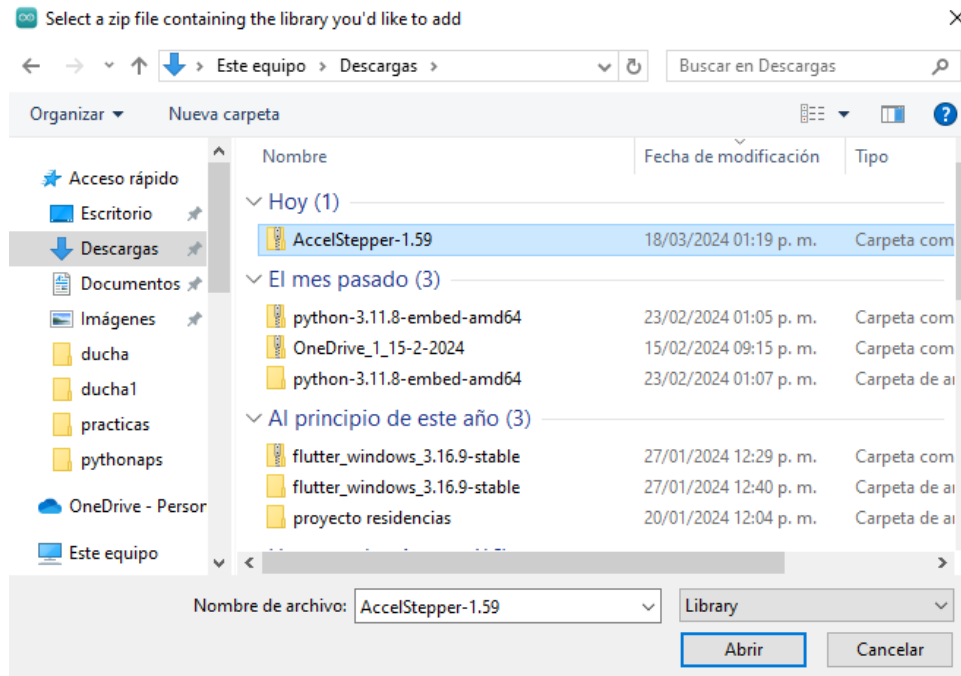




*Figura 101 Agregamos la librería descargada.*

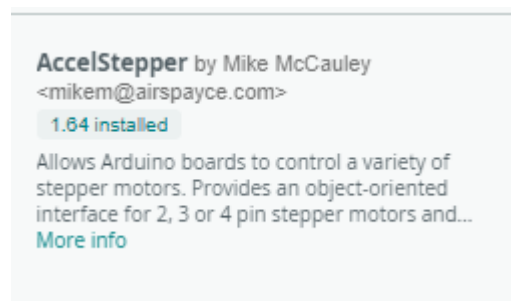
Abrimos la carpeta donde guardamos la carpeta .ZIP y la seleccionamos como se muestra a continuación en la figura 102.





*Figura 102 Abrimos nuestra carpeta de archivos y buscamos el .ZIP*

Ahora podemos ver que nuestra librería está instalada gracias al administrador de librerías como se observa en la figura 103.



*Figura 103 Verificamos que se instale correctamente.*

Una vez hecho esto, podemos añadir el código y cargarlo al Arduino para ejecutarlo, para ver los resultados. En este caso, el código de las figuras 104 y 105 indica que ambos motores se mueven simultáneamente dando una vuelta en sentido horario y una vuelta en sentido antihorario.

```

1  #define dirPin 2
2  #define stepPin 3
3  #define dirPin1 4
4  #define stepPin1 5
5  #define stepsPerRevolution 200
6
7  void setup() {
8      // Declare pins as output:
9      pinMode(stepPin, OUTPUT);
10     pinMode(dirPin, OUTPUT);
11     pinMode(stepPin1, OUTPUT);
12     pinMode(dirPin1, OUTPUT);
13 }
14 void loop() {
15     // giro horario
16     digitalWrite(dirPin, HIGH);
17     digitalWrite(dirPin1, HIGH);
18     // 1 revolucion en sentido horario, despacio
19     for (int i=0; i < stepsPerRevolution; i++) {
20         digitalWrite(stepPin, HIGH);
21         digitalWrite(stepPin1, HIGH);
22         delayMicroseconds(2000);
23         digitalWrite(stepPin, LOW);
24         digitalWrite(stepPin1, LOW);
25         delayMicroseconds(2000);
26     }

```

*Figura 104 Parte 1 de código en IDE de Arduino UNO*

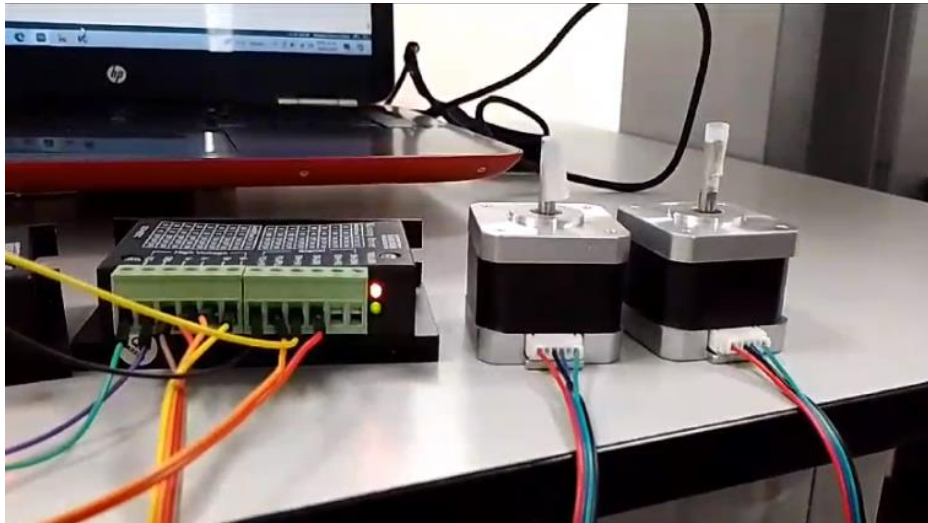
```

27     delay(1000);
28     // giros antihorarios
29     digitalWrite(dirPin, LOW);
30     digitalWrite(dirPin1, LOW);
31     // regresa el motor 1 revolucion en sentido antihorario, despacio
32     for (int i=0; i < stepsPerRevolution; i++) {
33         digitalWrite(stepPin, HIGH);
34         digitalWrite(stepPin1, HIGH);
35         delayMicroseconds(2000);
36         digitalWrite(stepPin, LOW);
37         digitalWrite(stepPin1, LOW);
38         delayMicroseconds(2000);
39     }
40     delay(1000);
41 }

```

*Figura 105 Parte 2 de código en IDE de Arduino UNO*

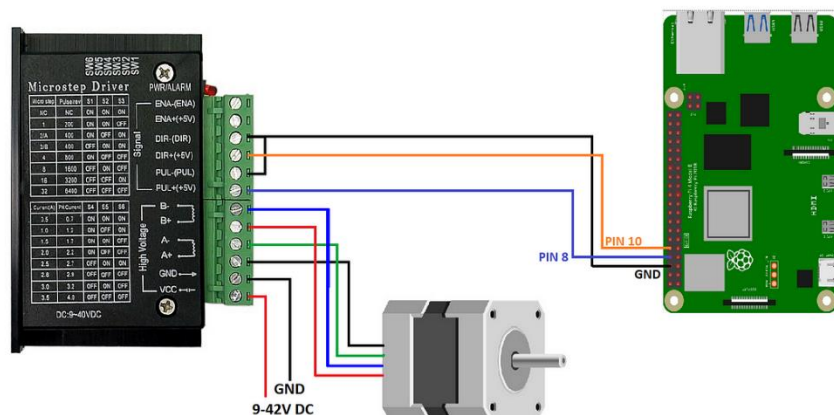
Una vez cargado el programase procede a ejecutar, en la figura 106 se muestra la ejecución de este programa.



*Figura 106 conexiones físicas de drivers y motores nema 17.*

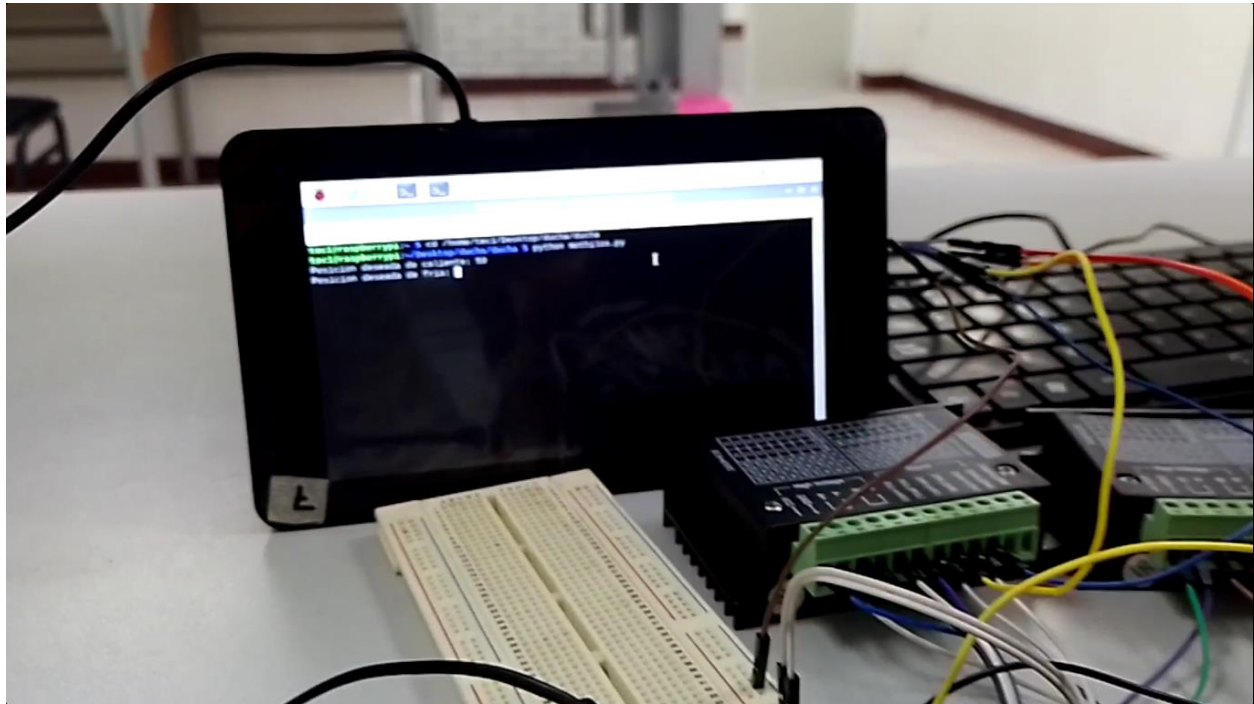
Para realizar la prueba preliminar de los motores NEMA 17 en la Raspberry Pi, es necesario realizar el diagrama esquemático de cada uno de los elementos que se integraran en el circuito, por lo tanto, se procederá a conectar la Raspberry Pi los controladores TB6600 de manera similar a como se muestra en la figura 107:

Diagrama esquemático:

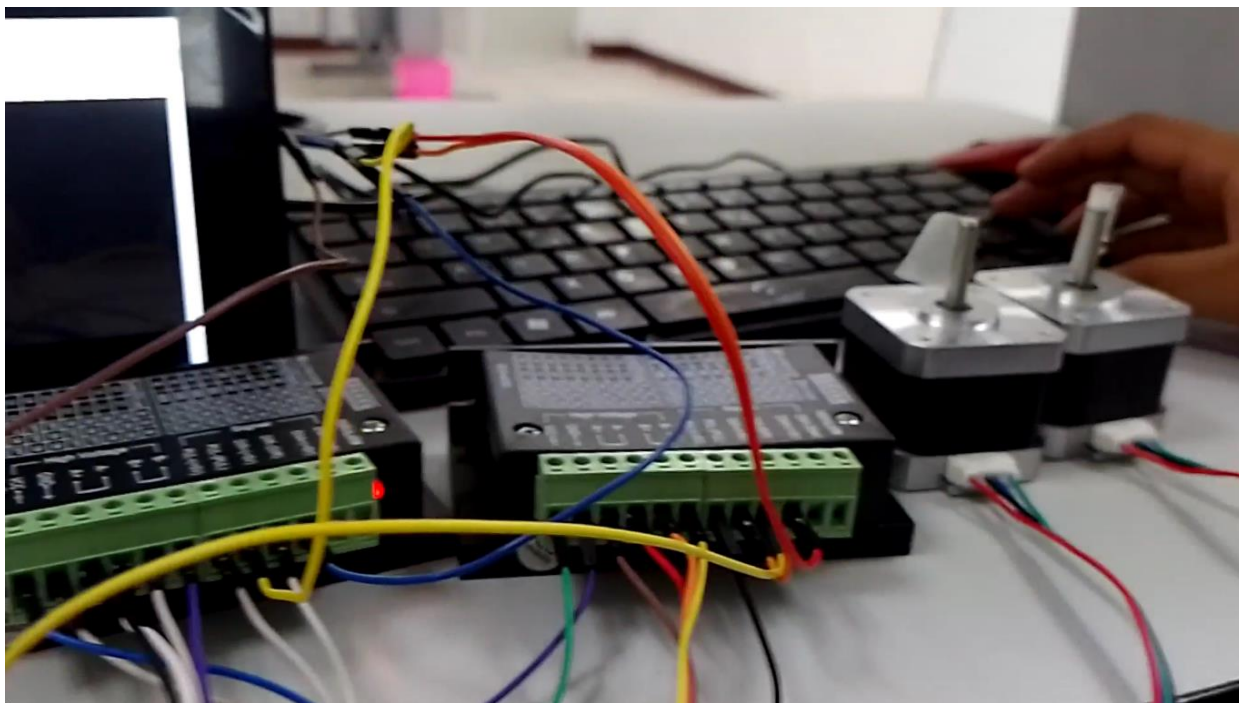


*Figura 107 Diagrama esquemático de Raspberry pi, drivers, motores nema 17.*

Mediante el diagrama esquemático, realizamos las conexiones necesarias tal como se muestra en la figura 108 y 109, utilizando la misma configuración para los controladores TB660 que en la prueba análoga de Arduino.



*Figura 108 Conexiones de Raspberry pi y drivers.*



*Figura 109 Conexión de drivers a motores nema 17.*

Para el desarrollo de esta prueba preliminar en la Raspberry, es necesario instalar la librería **RPi.GPIO**. Para lo cual utilizamos el comando **"pip install RPi.GPIO"** como se muestra a continuación en la figura 110.

```
tec1@raspberrypi:~$ pip install RPi.GPIO
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: RPi.GPIO in /usr/local/lib/python3.9/dist-packages
(0.7.1)
tec1@raspberrypi:~$
```

*Figura 110 Instalación de librerías.*

Una vez instaladas las librerías, podemos integrar el código en Visual Studio, mostrado en las figuras 111 a la 114, para obtener el archivo con extensión “.py” que transferiremos a la Raspberry Pi. Dicho código ha sido realizado de acuerdo con las necesidades del proyecto, pues el motor se mueve en posiciones específicas, que, al ser integrado al resto del código del sistema general, son dadas por el control difuso.

```
import RPi.GPIO as GPIO
import time
import threading

step_pincal = 17
dir_pincal = 27
step_pinfria = 23
dir_pinfria = 24

GPIO.setmode(GPIO.BCM)
GPIO.setup(step_pincal, GPIO.OUT)
GPIO.setup(dir_pincal, GPIO.OUT)
GPIO.setup(step_pinfria, GPIO.OUT)
GPIO.setup(dir_pinfria, GPIO.OUT)

def move_stepper1(steps, delay, clockwise_cal=True):
    if clockwise_cal:
        GPIO.output(dir_pincal, GPIO.HIGH)
        print(f"Motor caliente sentido horario.")
    else:
        GPIO.output(dir_pincal, GPIO.LOW)
        print(f"Motor caliente sentido antihorario.")
    print(f"Caliente movio {move_c} pasos, estas en la posicion {steps_c}.")
    for _ in range(steps):
        GPIO.output(step_pincal, GPIO.HIGH)
        time.sleep(delay)
        GPIO.output(step_pincal, GPIO.LOW)
        time.sleep(delay)
```

*Figura 111 Parte 1 de código en Visual Studio.*

```
def move_stepper2(steps, delay, clockwise_fria=True):
    if clockwise_fria:
        GPIO.output(dir_pinfria, GPIO.HIGH)
        print(f"Motor fria sentido horario.")
    else:
        GPIO.output(dir_pinfria, GPIO.LOW)
        print(f"Motor caliente sentido antihorario.")
    print(f"Fria movio {move_f} pasos, estas en la posicion {steps_f}.")
    for _ in range(steps):
        GPIO.output(step_pinfria, GPIO.HIGH)
        time.sleep(delay)
        GPIO.output(step_pinfria, GPIO.LOW)
        time.sleep(delay)
```

*Figura 112 Parte 2 de código en Visual Studio.*



```

try:
    prev_c = 0
    move_c = 0
    prev_f = 0
    move_f = 0
    while True:
        # entradas del usuario, caliente y frio
        steps_input_c = input("Posicion deseada de caliente: ")
        steps_c = int(steps_input_c)
        steps_input_f = input("Posicion deseada de fria: ")
        steps_f = int(steps_input_f)
        move_c = abs(steps_c - prev_c)
        move_f = abs(steps_f - prev_f)
        if (steps_c >= 0 and steps_c <= 200 and steps_f >= 0 and steps_f <= 200):
            # uso de hilos para mover simultáneamente los motores caliente y frio
            thread1 = threading.Thread(target=move_stepper1, args=(move_c, 0.01, steps_c >= prev_c))#move por steps
            thread2 = threading.Thread(target=move_stepper2, args=(move_f, 0.01, steps_f >= prev_f))#move por steps
            thread1.start()
            thread2.start()
            thread1.join()
            thread2.join()

            # Actualizar las posiciones previas
            prev_c = steps_c
            prev_f = steps_f
        else:
            print(f"Tus {steps_c}y{steps_f} pasos superan el movimiento. Te encuentras en el paso {prev_c}y{prev_f}.")

```

Figura 113 Parte 3 de código en Visual Studio.

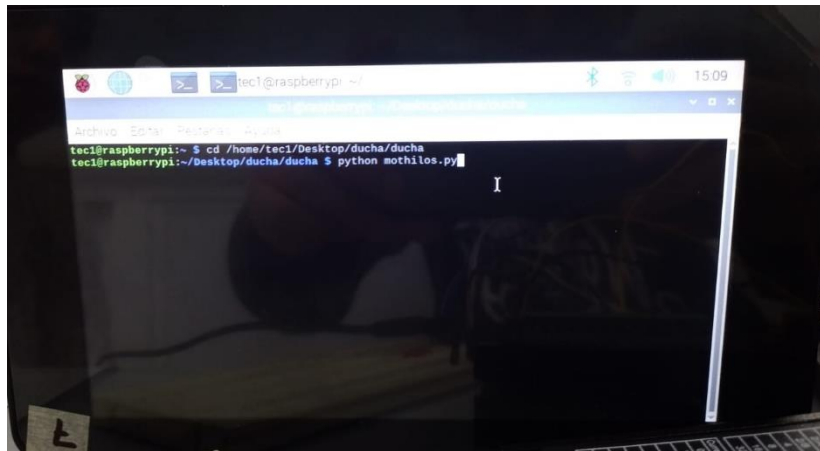
```

except ValueError as ve:
    print(f"Error: {ve}. Escriba un número por favor.")
finally:
    # Limpiar pines GPIO al finalizar
    GPIO.cleanup()

```

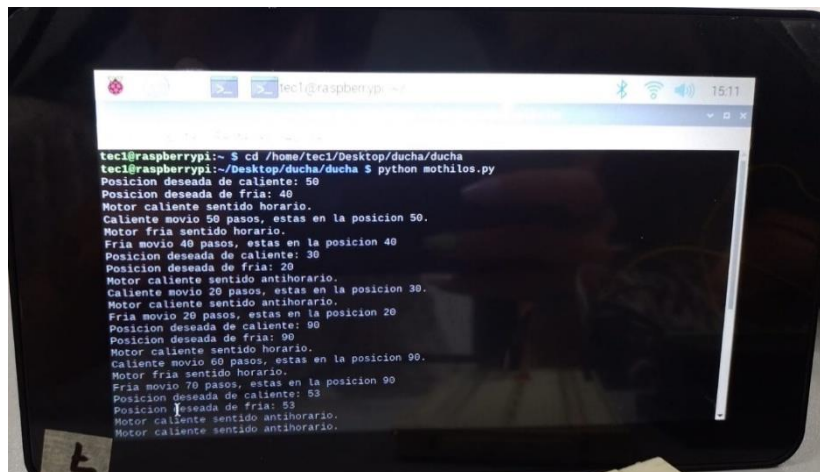
Figura 114 Parte 4 de código en Visual Studio.

Para utilizar dicho código en la Raspberry pi, es necesario ejecutarlo desde la terminal, mediante los comandos de cambio de directorio a la carpeta donde se encuentra el archivo con extensión .py **"cd /home/tec1/Desktop/ducha/ducha"** y el de ejecución **"python motorhilos.py"** como se muestra en la figura 115



*Figura 115 Cambio de directorio a la carpeta donde está el archivo .py*

Una vez ejecutado el código, la terminal de la Raspberry pi indica la posición en que se encuentra cada motor NEMA 17 y los pasos que tuvo que avanzar para llegar a ella, además del sentido en el que giró, dicha terminal se muestra en la figura 116.



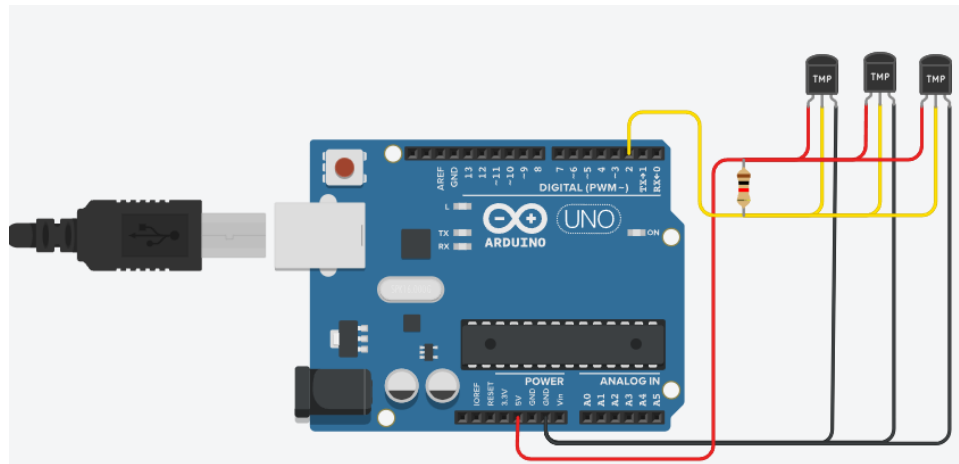
*Figura 116 Ejecución del código.*

### 3.5.3 Pruebas unitarias de integración para sensores DS18B20

La prueba preliminar de los sensores de temperatura DS18B20 en Arduino UNO, como las anteriores, se realizó para comprobar el funcionamiento correcto de los sensores. Desarrollando un código previo al utilizado en el sistema que integra al proyecto.

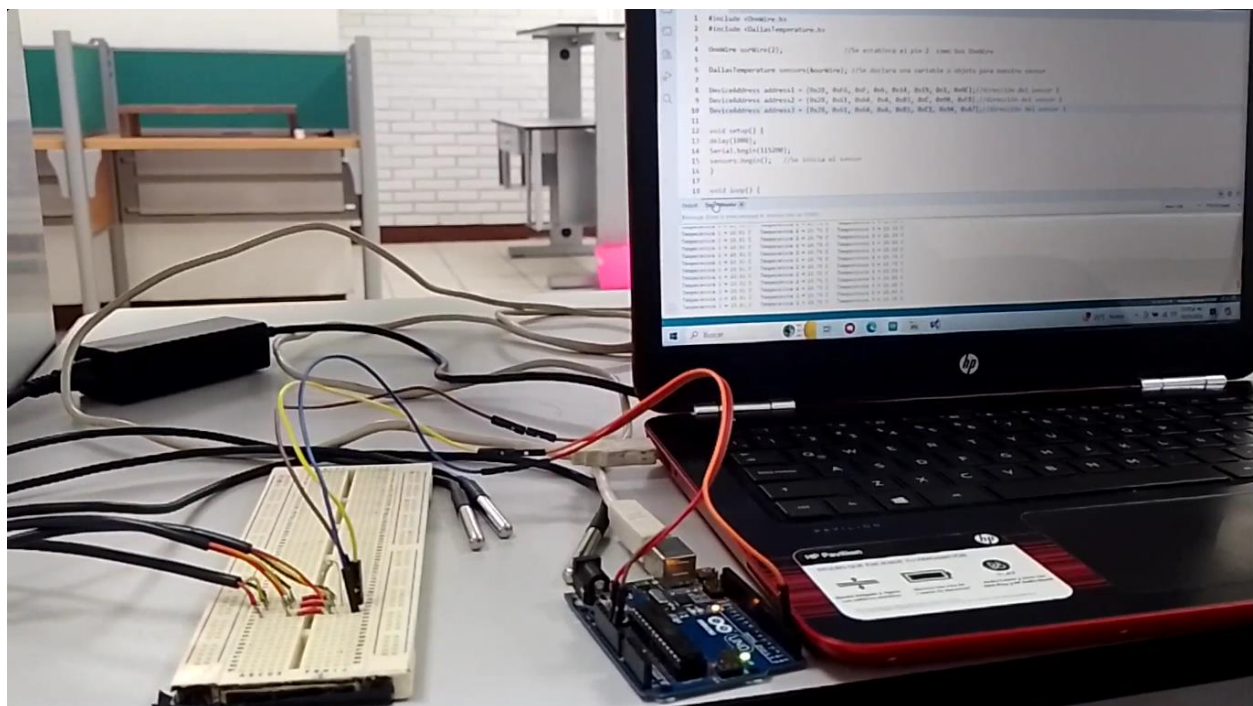


Para ello, se realizó la conexión de los sensores en el diagrama esquemático de la figura 117.



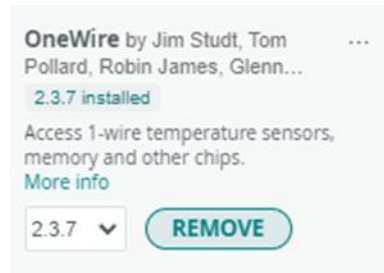
*Figura 117 Diagrama esquemático de Arduino UNO y sensores DS18b20.*

Posteriormente se realizaron las conexiones correspondientes de manera física entre el Arduino Uno y los sensores de temperatura DS18B20, considerando la resistencia de 4.7k Ohms del diagrama.

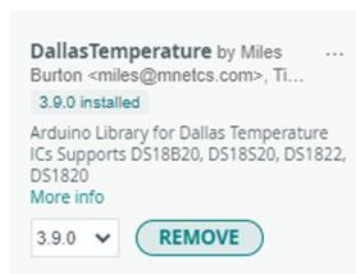


*Figura 118 conexiones de Arduino UNO a sensores DS18B20.*

En el entorno de Arduino, es necesario tener instaladas las librerías necesarias, en este caso, **OneWire** y **DallasTemperature** como se muestra en las imágenes 119 y 120. La primera librería permite el uso de un solo cable para la transferencia de datos, pues cada sensor tiene un identificador unico que permite a los microcontroladores identificar de que sensor se recibe la información e interpretarla por separado. Por otra parte, la segunda librería se utiliza para que el microcontrolador pueda interpretar los datos obtenidos por cada sensor



*Figura 119 Instalación de librería OneWire.*



*Figura 120 Instalación de librería DallasTermperature.*

En esta versión implementada en Arduino, es necesario realizar 2 códigos, o unirlos en uno solo, el primero se utiliza para obtener los identificadores de cada sensor y en el segundo, con dichos identificadores, podemos realizar las mediciones de temperatura como se muestra en la figura 121.

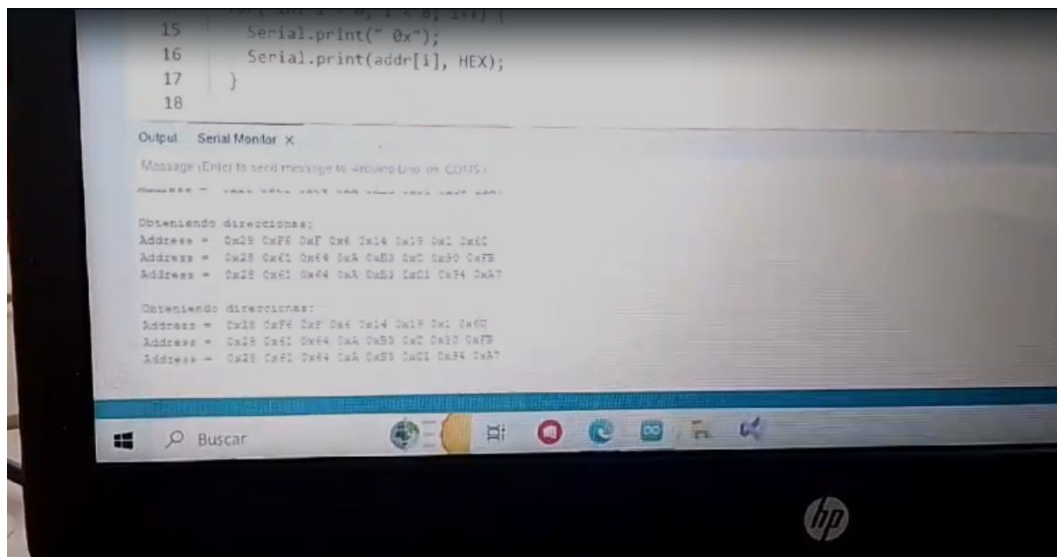
```

1  #include <OneWire.h>
2
3  OneWire ourWire(2);           //Se establece el pin 2 como bus OneWire
4
5  void setup(void) {
6      Serial.begin(11520);
7  }
8
9  void loop(void) {
10     byte addr[8];
11     Serial.println("Obteniendo direcciones:");
12     while (ourWire.search(addr))
13     {
14         Serial.print("Address = ");
15         for( int i = 0; i < 8; i++) {
16             Serial.print(" 0x");
17             Serial.print(addr[i], HEX);
18         }
19         Serial.println();
20     }
21
22     Serial.println();
23     ourWire.reset_search();
24     delay(2000);
25 }

```

*Figura 121 Código 1 para identificar los sensores.*

Una vez conectados los sensores al mismo pin del Arduino Uno y alimentados correctamente, como se muestra en el diagrama de conexiones, podemos cargar el código al Arduino. Al iniciar el monitor serial, comenzaran a aparecer las direcciones de cada sensor en código hexadecimal como se muestra en la figura 122.



*Figura 122 Direcciones de los 3 sensores.*

El segundo código, disponible en la figura 123, es completado con los identificadores de cada sensor y permite la visualización de la temperatura de cada sensor.

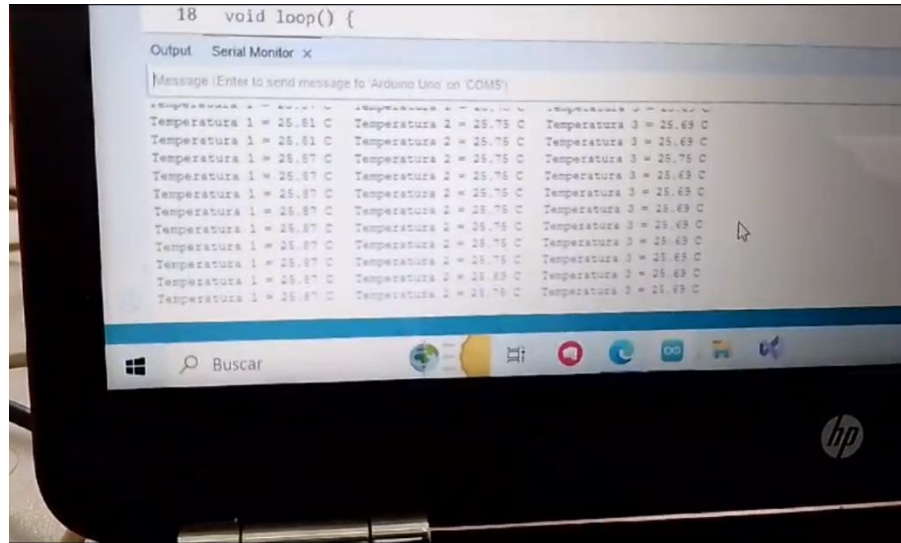
```

1  #include <OneWire.h>
2  #include <DallasTemperature.h>
3
4  OneWire ourWire(2);           //Se establece el pin 2 como bus OneWire
5
6  DallasTemperature sensors(&ourWire); //Se declara una variable u objeto para nuestro sensor
7
8  DeviceAddress address1 = {0x28, 0xFF, 0xCA, 0x4A, 0x5, 0x16, 0x3, 0xBD}; //dirección del sensor 1
9  DeviceAddress address2 = {0x28, 0xFF, 0x89, 0x3A, 0x1, 0x16, 0x4, 0xAF}; //dirección del sensor 2
10 DeviceAddress address3 = {0x28, 0xFF, 0x23, 0x19, 0x1, 0x16, 0x4, 0xD9}; //dirección del sensor 3
11
12 void setup() {
13     delay(1000);
14     Serial.begin(9600);
15     sensors.begin(); //Se inicia el sensor
16 }
17
18 void loop() {
19
20     sensors.requestTemperatures(); //envía el comando para obtener las temperaturas
21     float temp1= sensors.getTempC(address1); //Se obtiene la temperatura en °C del sensor 1
22     float temp2= sensors.getTempC(address2); //Se obtiene la temperatura en °C del sensor 2
23     float temp3= sensors.getTempC(address3); //Se obtiene la temperatura en °C del sensor 3
24
25     Serial.print("Temperatura 1 = ");
26     Serial.print(temp1);
27     Serial.print(" C");
28     Serial.print(" Temperatura 2 = ");
29     Serial.print(temp2);
30     Serial.print(" C");
31     Serial.print(" Temperatura 3 = ");
32     Serial.print(temp3);
33     Serial.println(" C");
34     delay(100);
35     delay(100);
36 }

```

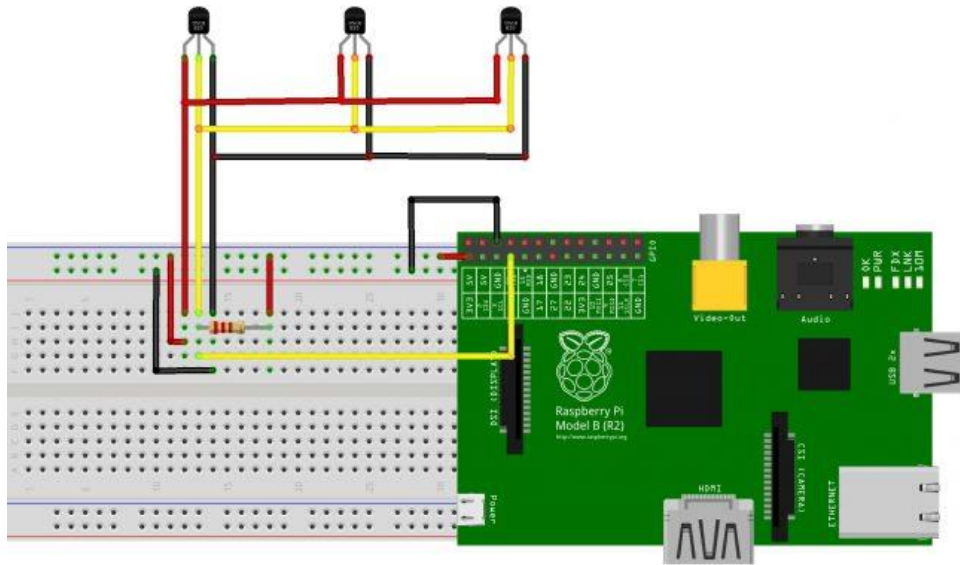
*Figura 123 Identificadores para poder medir la temperatura.*

Con las mismas conexiones en los sensores, cargamos el código al Arduino UNO y al comenzar la visualización del monitor serial, los valores de temperatura medidos por los sensores DS18B20 comienzan a mostrarse, como se aprecia en la figura 124, por lo cual, podemos realizar variaciones individuales en los sensores mediante la fuente de calor.



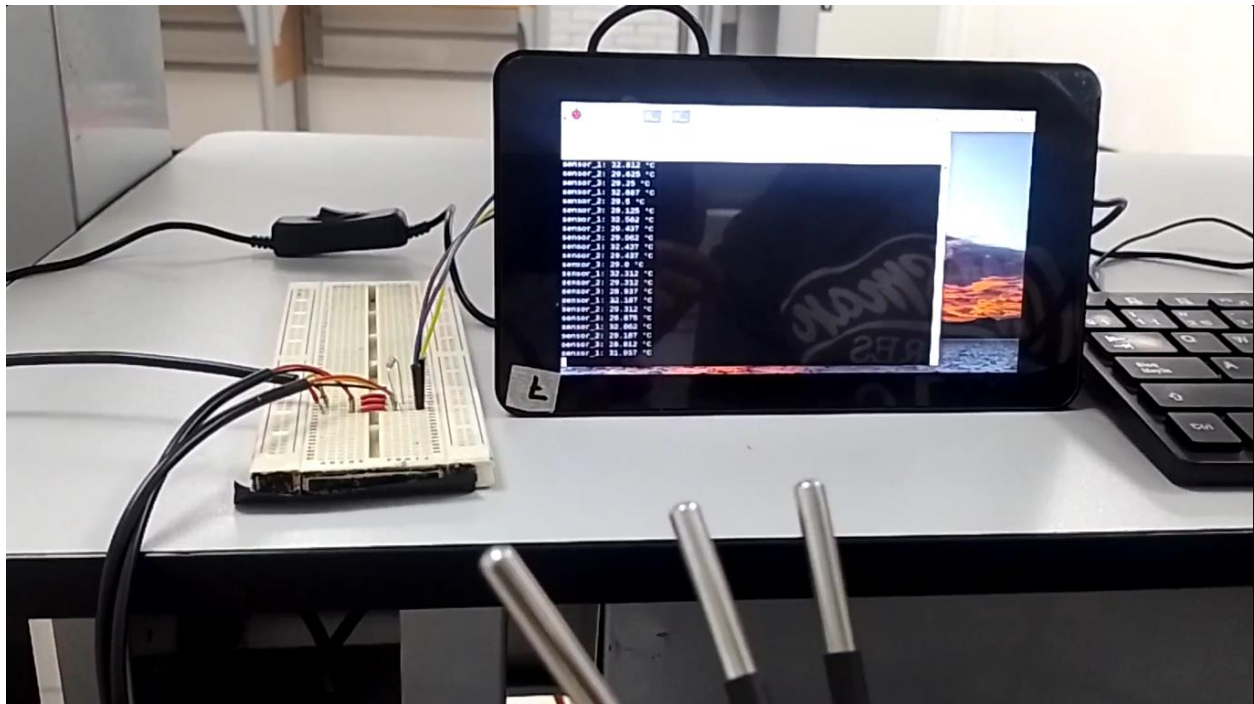
*Figura 124 Dirección de sensores y temperaturas registradas.*

Al momento de realizar la prueba preliminar de los sensores de temperatura DS18B2 en la Raspberry Pi, se diseña el diagrama esquemático con cada uno de los elementos que se integrarán en el circuito, por lo tanto, se procederán a conectar los sensores de temperatura a la Raspberry Pi como se muestra a continuación en la figura 125



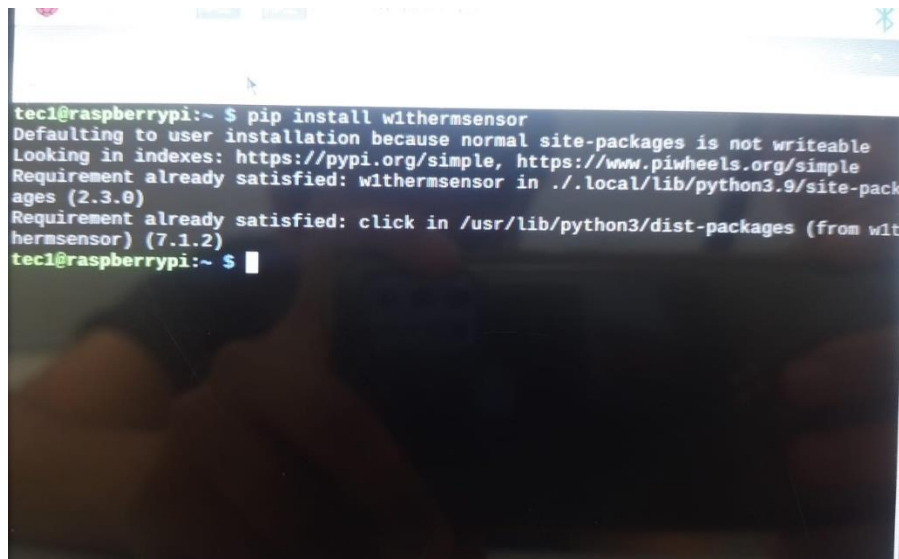
*Figura 125 Diagrama esquemático de sensores y Raspberry Pi.*

Realizamos las conexiones físicas de acuerdo con el diagrama de conexiones anterior como en la figura 126.



*Figura 126 Conexiones de sensores y Raspberry Pi.*

Para el desarrollo de esta prueba preliminar en el entorno de la Raspberry pi 4, es necesario tener instaladas las librerías necesarias, en este caso, **w1thermsensor**.

A terminal window on a Raspberry Pi. The prompt is 'tecl@raspberrypi:~'. The user enters 'pip install w1thermsensor'. The output shows that the requirement is already satisfied in the local site-packages (2.3.0) and the click requirement is also satisfied in the dist-packages (7.1.2).

```
tecl@raspberrypi:~ $ pip install w1thermsensor
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Requirement already satisfied: w1thermsensor in ~/.local/lib/python3.9/site-packages (2.3.0)
Requirement already satisfied: click in /usr/lib/python3/dist-packages (from w1thermsensor) (7.1.2)
tecl@raspberrypi:~ $
```

*Figura 127 Instalación de librerías.*

Otro paso importante es habilitar la interfaz **OneWire** en la Raspberry Pi, mediante el archivo de configuración config.txt. Para ello es necesario abrir una terminal en la Raspberry Pi y ejecutar el siguiente comando para editar el archivo de configuración:

```
sudo nano /boot/config.txt
```

*Figura 128 Comando a ejecutar.*

A terminal window on a Raspberry Pi. The prompt is 'tecl@raspberrypi:~'. The user enters 'sudo nano /boot/config.txt'.

```
tecl@raspberrypi:~ $ sudo nano /boot/config.txt
```

*Figura 129 Comando en Raspberry Pi.*

En dicho archivo es necesario editar la última línea con el enunciado **“dtoverlay=w1-gpio,gpiopin=4”** como se muestra en la figura 130 y guardar los cambios en dicho archivo presionando **“Ctrl + O”**. Esto definirá al pin GPIO 4 como el pin que recibirá el bus de los sensores mediante el uso de **One Wire**.



```
disable_overscan=1

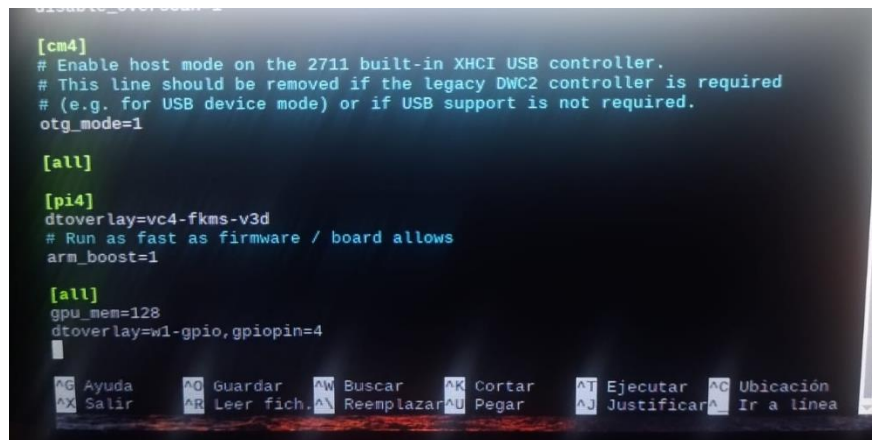
[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[all]

[pi4]
dtoverlay=vc4-fkms-v3d
# Run as fast as firmware / board allows
arm_boost=1

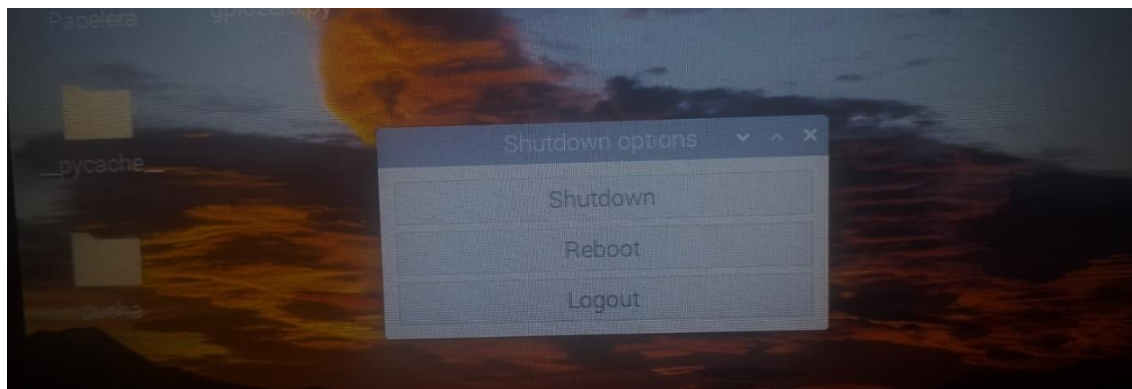
[all]
gpu_mem=128
dtoverlay=w1-gpio,gpiopin=4

```



*Figura 130 Cambios en el archivo.*

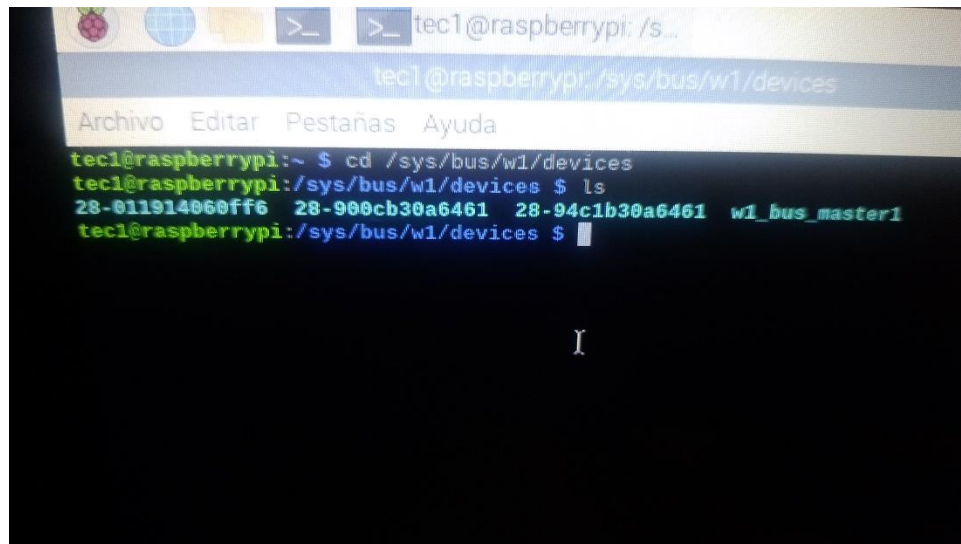
Después se debe reiniciar la Raspberry para que se apliquen los cambios, como se muestra de manera ilustrativa en la figura 131.



*Figura 131 Reinicio de la Raspberry Pi.*

Una vez completados los pasos anteriores, y teniendo conectados los sensores, se ejecutan los siguientes comandos para conocer los dispositivos DS18B20 conectados en el bus One Wire. El primero es un cambio de directorio mediante **"cd /sys/bus/w1/devices/"** y el segundo es la búsqueda de dispositivos mediante **"ls"**.

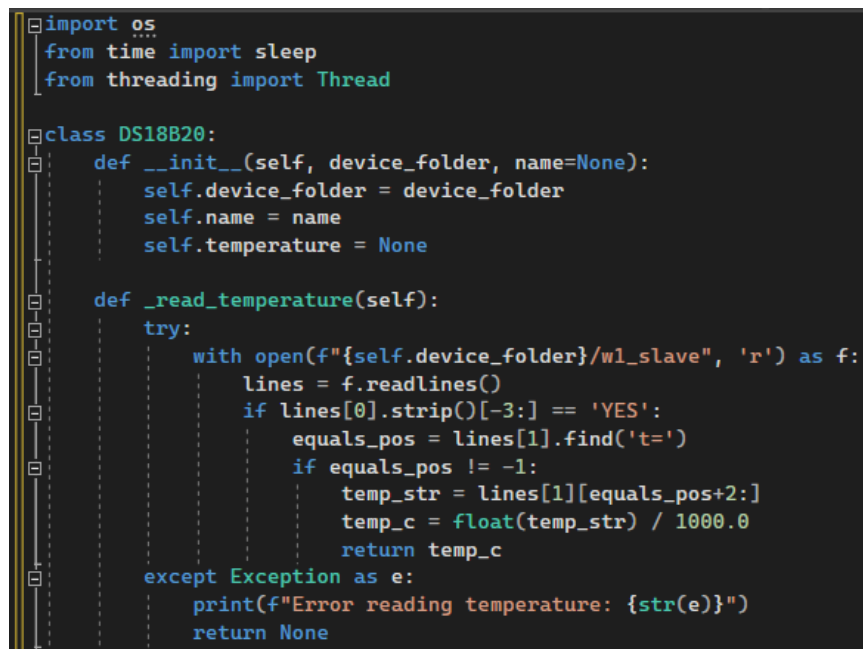




```
tec1@raspberrypi: /s...
tec1@raspberrypi: /sys/bus/w1/devices
Archivo  Editar  Pestañas  Ayuda
tec1@raspberrypi:~ $ cd /sys/bus/w1/devices
tec1@raspberrypi:/sys/bus/w1/devices $ ls
28-011914060ff6 28-900cb30a6461 28-94c1b30a6461 w1_bus_master1
tec1@raspberrypi:/sys/bus/w1/devices $
```

*Figura 132 Búsqueda de dispositivos.*

Dichos dispositivos deben añadirse a nuestro código en la sección de definir las direcciones de los sensores DS18B20. Una vez realizado este proceso, el código se visualiza de la siguiente manera dentro de las figuras 133 a la 135.



```
import os
from time import sleep
from threading import Thread

class DS18B20:
    def __init__(self, device_folder, name=None):
        self.device_folder = device_folder
        self.name = name
        self.temperature = None

    def _read_temperature(self):
        try:
            with open(f"{self.device_folder}/w1_slave", 'r') as f:
                lines = f.readlines()
                if lines[0].strip()[-3:] == 'YES':
                    equals_pos = lines[1].find('t=')
                    if equals_pos != -1:
                        temp_str = lines[1][equals_pos+2:]
                        temp_c = float(temp_str) / 1000.0
                        return temp_c
        except Exception as e:
            print(f"Error reading temperature: {str(e)}")
            return None
```

*Figura 133 Parte 1 de código para agregar las direcciones de los sensores..*

```

def update_temperature(self):
    temp = self._read_temperature()
    if temp is not None:
        self.temperature = temp

def read_temperatures(sensors):
    try:
        while True:
            for sensor in sensors:
                sensor.update_temperature()
                print(f"{sensor.name}: {sensor.temperature} °C")
            sleep(1)
    except KeyboardInterrupt:
        print("Programa terminado por el usuario.")

if __name__ == '__main__':
    # Definir las direcciones de los sensores DS18B20
    sensor_addresses = [
        "/sys/bus/w1/devices/28-011914060ff6",
        "/sys/bus/w1/devices/28-900cb30a6461",
        "/sys/bus/w1/devices/28-94c1b30a6461"
    ]

```

Figura 134 Parte 2 de código con las direcciones de los sensores.

```

# Crear instancias de sensores DS18B20
sensors = [DS18B20(address, name=f"sensor_{i+1}") for i, address in enumerate(sensor_addresses)]

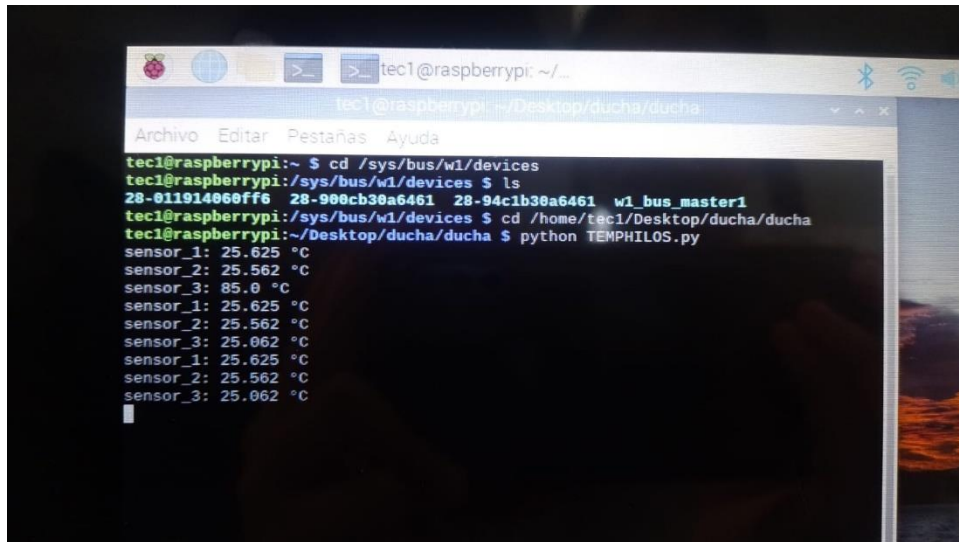
# Crear e iniciar el hilo para la lectura de temperaturas
temperature_thread = Thread(target=read_temperatures, args=(sensors,))
temperature_thread.start()

try:
    temperature_thread.join() # Esperar a que el hilo termine (Ctrl+C para detener)
except KeyboardInterrupt:
    print("Programa terminado por el usuario.")
finally:
    # Limpiar GPIO
    GPIO.cleanup()

```

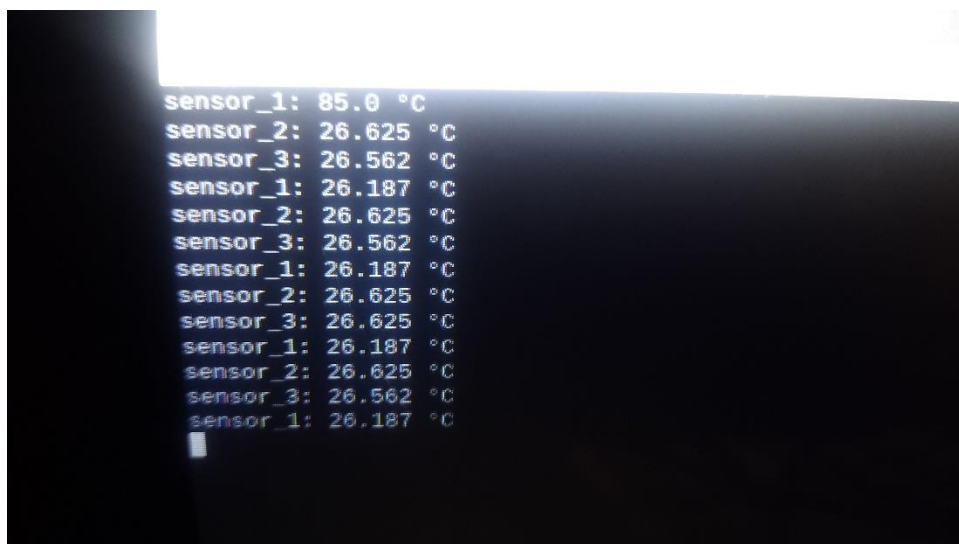
Figura 135 Parte 3 de código de los sensores en Visual Studio en lenguaje Python.

Con el código en Python, obtenemos el archivo con extensión **“.py”**, en este caso lo nombramos **“TEMPHILOS.py”**, y lo colocamos dentro de una carpeta en la Raspberry. Dentro de la terminal de la Raspberry pi (figura 136), ejecutamos los siguientes comandos, cambio de directorio a la carpeta donde se encuentra nuestro código, en este caso **“cd /home/tec1/Desktop/ducha/ducha”**, y para ejecutar el archivo escribimos **“python TEMPHILOS.py”**

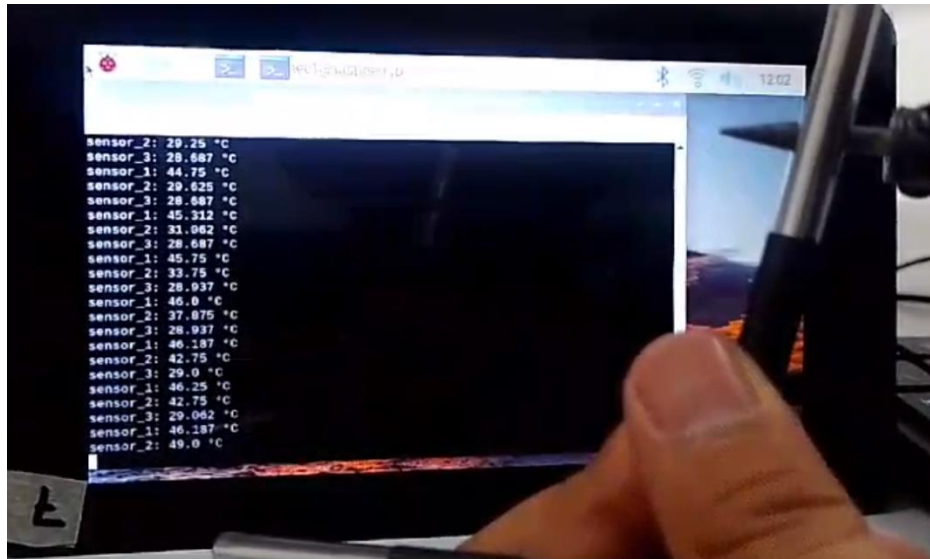


*Figura 136 Cambio de directorio a donde se encuentra la carpeta.*

En la figura 137 se muestran algunas mediciones de temperatura realizadas con los 3 sensores DS18B20 dentro del entorno de la Raspberry pi y en la figura 138 se muestra la variación en las mediciones al utilizar una fuente de calor, en este caso un cañón.



*Figura 137 Mediciones de los sensores usando una fuente de calor.*



*Figura 138 Variación de fuente de calor en los sensores y sus respectivas temperaturas.*

### **3.6 Etapa de implementación.**

La etapa de implementación se centra en reunir los avances realizados durante las etapas de diseño y las pruebas preliminares dentro de un prototipo físico funcional, haciendo adaptaciones a los códigos utilizados, montando las piezas necesarias para el prototipo. Se hace uso del software desarrollado hasta el momento con el objetivo de crear un sistema capaz de regularse a sí mismo a partir de los requerimientos del usuario y los datos de los sensores.

#### **3.6.1 Dimensionamiento del prototipo físico.**

Dentro del dimensionamiento del prototipo físico se consideraron las medidas obtenidas en el modelo 3D del prototipo, pues las medidas mostradas guardan relación 1:1 con piezas del mundo real. Aun con ello presente, se dejó un margen de error de 0.5 cm para cada pieza, con el fin de evitar posibles problemas de acomodamiento y facilitar su instalación.

Para realizar las piezas del prototipo se utilizó una hoja de MDF con espesor de 9 milímetros observada en la figura 139.



*Figura 139 MDF 9mm.*

Con dicha hoja de MDF, se trazaron las medidas sobre el MDF, basándonos en el modelo 3D y obteniendo segmentos rectangulares que fueron posteriormente cortados como se observa en la figura 140. Con el fin de aprovechar mejor las partes rectangulares y evitar crear desperdicios de madera, las piezas más pequeñas del prototipo fueron obtenidas de los laterales del rectángulo más grande, dejando el menor número de desperdicios de madera (figura 141).



*Figura 140 Corte de sección en MDF.*



*Figura 141 Corte de las piezas.*

### 3.7 Creación del prototipo con el sistema hidráulico y electrónico.

#### 3.7.1 Ensamblado del sistema hidráulico y electrónico.

Una vez cortadas todas las piezas de MDF se unieron mediante pegamento blanco y clavos, observados en las imágenes 142 y 143 de acuerdo con el esquema realizado en 3D del prototipo, dando como resultado, la estructura de la figura 144.



*Figura 142 Clavos de una pulgada.*



*Figura 143 Pegamento blanco 850.*





*Figura 144 Estructura de MDF.*



Una vez cortados todos los tubos de CCPVC se lijaron con una lija como la de la figura 145, de grano 100, por la parte interna (los elementos tee de CCPVC y codos del CCPVC) como se muestra en la figura 146, mientras que a los tubos de CCPVC se lijaron por la parte externa, como en la figura 147, para que el cemento se adhiriera mejor entre cada elemento y para quitar las rebabas producidas por el corte.



*Figura 145 Lija de grano 100*



*Figura 146 Pieza sutilmente lijada de la parte interna.*



*Figura 147 Pieza sutilmente lijada de la parte externa.*

Una vez cortados y lijados los tubos de CCPVC procedimos a verificar que las medidas fueran exactas en ambos lados del sistema hidráulico como se muestra en la figura 148. Además, se revisó que encajaran dentro del cuerpo de la estructura de MDF, de la manera en que se ilustra en la figura 149.



*Figura 148 Comprobación de las medidas de a las medidas reales de cada elemento.*



*Figura 149 Comprobación de medidas en estructura de MDF.*

Para la unión de cada uno los elementos se les colocó cemento para CCPVC, esto evitará filtraciones de agua, por lo cual se esperaron 24 horas para poder utilizar las piezas.

Del lado derecho del sistema hidráulico se posicionaron las piezas de la siguiente manera válvula como se muestra en la figura 150, caudalímetro en las figuras 151 Y 152.



*Figura 150 válvula de paso con conexiones de tubo y codo.*



*Figura 151 Conexión de caudalímetro con tubos y conexiones necesarias.*



*Figura 152 Conexión de ambos lados del caudalímetro.*

Además, se realizaron los orificios de acuerdo con el diámetro del sensor DS18B20 para introducirlo dentro del sistema de tubería por medio de una Tee de CCPVC que fue sellada previamente con un tapón para tuberías de 1/2" como se muestra en la figura 153.



*Figura 153 Orificio en el tapón donde ira el sensor DS18B20.*

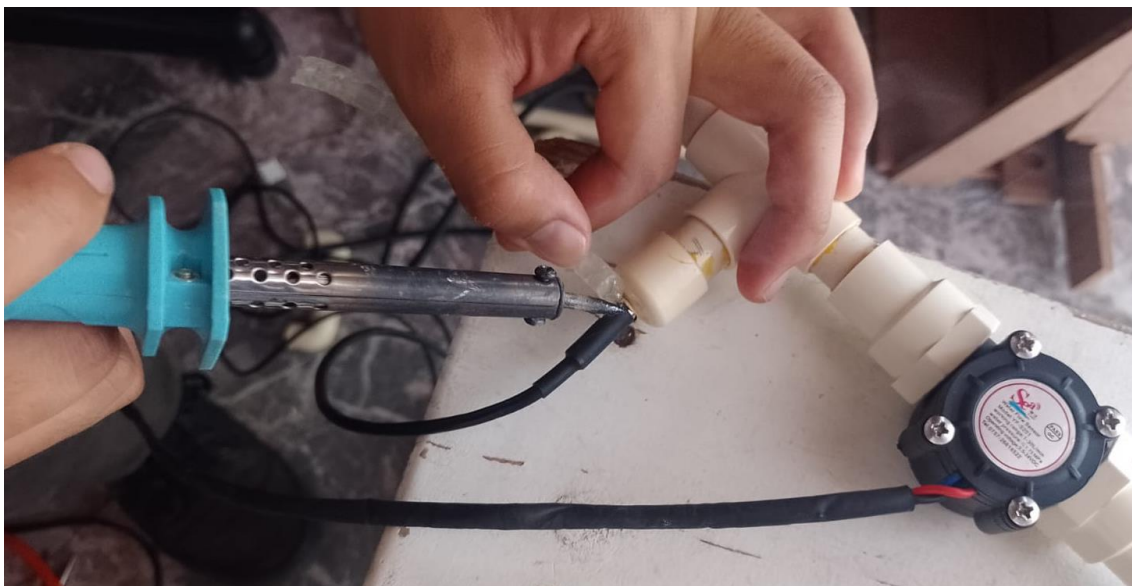


Posterior a la realización del orificio ingresamos el sensor en él como se muestra en figura 154.



*Figura 154 Ingreso del sensor DS18B20 en el orificio*

También se aisló el orificio con silicón para la prevención de fugas o filtrado de agua como se muestra en la figura 155, resultando en la pieza de la figura 156.



*Figura 155 Aislamiento del orificio del sensor DS18B20.*



*Figura 156 Sensor DS18B20 aislado correctamente.*

De ambos lados del sistema hidráulico se posicionaron las piezas de la siguiente manera:

Caudalímetro como se muestra en la figura 157.



*Figura 157 caudalímetro adherido correctamente.*

La válvula se unió junto a su codo y tubería de CCPVC con cemento como se muestra en la figura 158.





*Figura 158 Unión de la válvula a los tubos y codos con cemento.*

Sensor de temperatura con tapón hembra sellado junto a tubería CCPVC como se muestra en la figura 159.



*Figura 159 Sensor de temperatura en su pieza de tubería.*

Se proceden a unir las tres piezas mencionadas de ambos lados, pegándolas perfectamente como se muestra en la figura 160.



*Figura 160 Sistema hidráulico adherido correctamente.*

Los dos lados se unen de manera similar, pero intercambiando únicamente la válvula por el caudalímetro en el orden de posición, con el fin de permitir que el motor tenga espacio para colocarse. Posteriormente se unirán ambos lados del sistema hidráulico por medio de un bigote de CCPVC de 1/2" como se muestra en las figuras 161 a la 163.



*Figura 161 Bigote de CCPVC.*



*Figura 162 Se adhiere el lado derecho del sistema hidráulico.*



*Figura 163 Se adhiere el lado izquierdo del sistema hidráulico.*

En la figura 164 se muestra como en la parte superior se adhiere una tee de CCPVC con sensor de temperatura debidamente aislado y ya ensamblado como se indicó en pasos anteriores.



*Figura 164 Se adhiere la te de CCPVC que conecta al bigote con el sensor.*



De esta forma, una parte del sistema hidráulico y una parte del sistema electrónico se encuentran unidos como se muestra en la figura 165.



*Figura 165 Sistema hidráulico y electrónico adherido parcialmente.*

Una vez adheridos los elementos de cada uno del sistema hidráulico, se puede revisar su tamaño dentro de la estructura de MDF como se muestra en la figura 165, para evitar errores.



*Figura 166 Sistema adherido a la estructura de MDF.*

Para la terminación de unir del sistema hidráulico con el sistema electrónico se realizó la integración del motor NEMA 17 con la válvula por medio de la herramienta denominada tarraja. Para poder realizar la cuerda al motor se usó una tarraja M5 x 0.8 mostrada en la figura 167 y la misma tarraja al lado del motor en la figura 168.



*Figura 167 Tarraja M5 de 0.8mm*



*Figura 168 Tarraja y motor nema 17.*

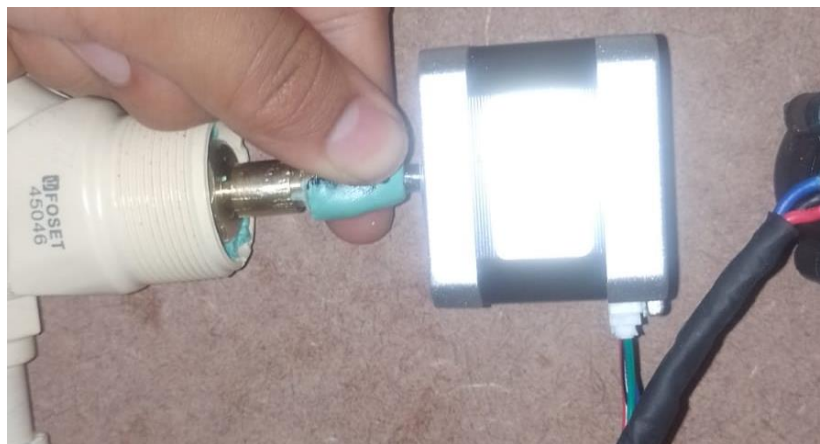


Se procede a colocar la tarraja encima del husillo de ambos motores NEMA 17 y girarla en sentido de las manecillas de reloj como se muestra en la figura 169.



*Figura 169 Tarraja colocada para girar.*

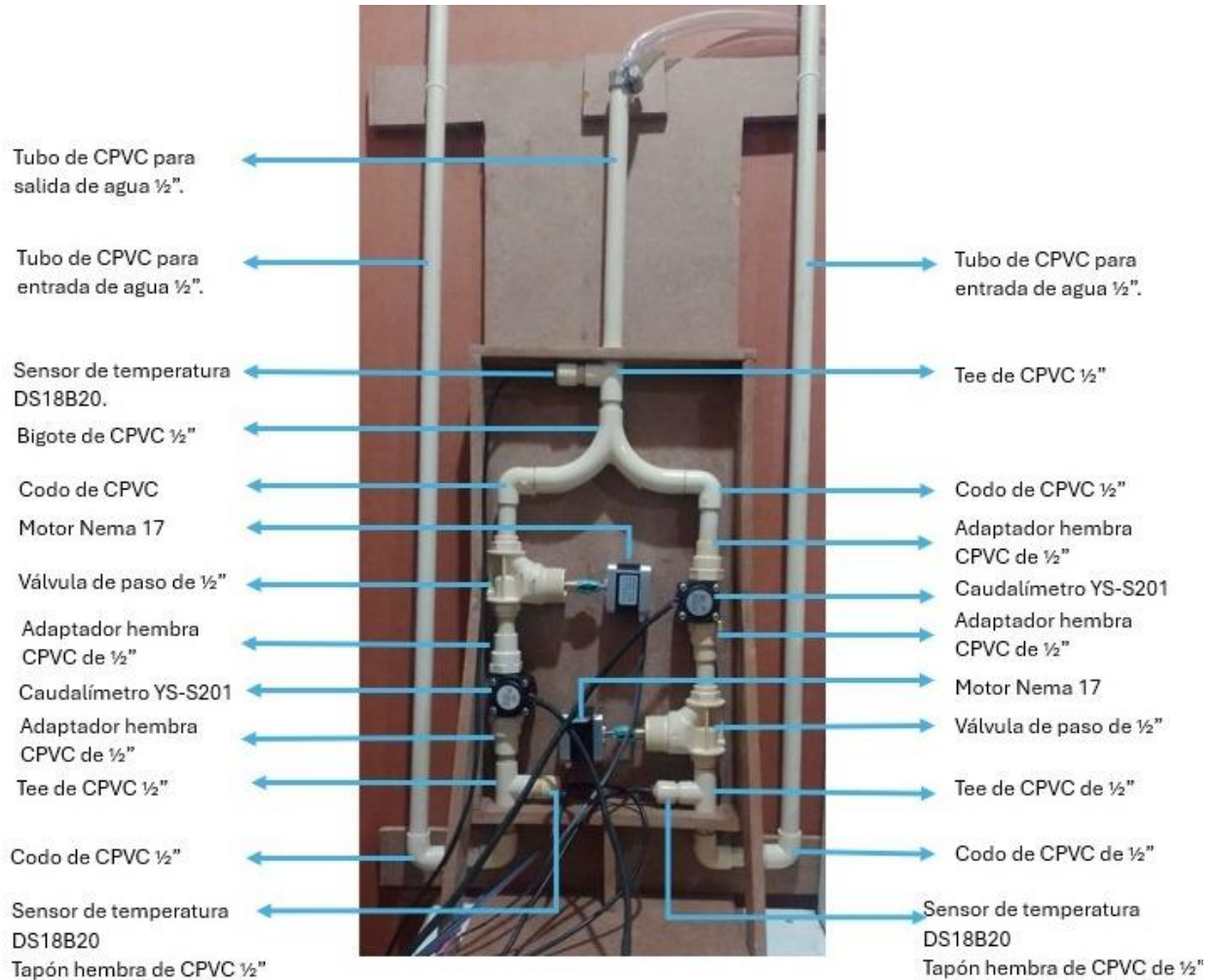
Al tener el motor NEMA 17 la cuerda adecuada para entrar en el roscado de la válvula se procede a unir la válvula y el motor NEMA 17 como se muestra en la figura 170, primero introduciendo el husillo en el roscado y después asegurando la unión con masilla epódica de dos componentes.



*Figura 170 Acoplamiento de válvula y motor nema.*



Tras realizar las uniones de ambos motores, el sistema hidráulico – electrónico queda completado y puede colocarse dentro de la estructura del MDF, completándolo con los tubos de alimentación de agua caliente y fría, obteniendo el sistema tal y como se observa en la figura 171.



*Figura 171 Sistema hidráulico y electrónico finalizado.*

## Capítulo IV. Resultados

### 4.1 Simulación de control difuso con Matlab

Dentro de las simulaciones del control difuso realizadas en Matlab, nos enfocamos en utilizar principalmente a las variables que utilizará dicho control dentro del sistema, considerando como entradas a “valor\_deseado” y “valor\_sensado”, de las cuales, se obtiene su diferencia y a ese valor lo nombramos “error”. Dicho error lo fuzzificamos dentro del control difuso, considerando valores desde -70 hasta 70, pues se en algunas simulaciones, el error podría ser demasiado alto si se consideran temperaturas muy separadas entre sí, de forma que si llega a haber un error en el sensor de temperatura de la mezcla (valor\_sensado) y tenga un valor muy bajo, igual o menor a 0°C, o muy alto, cercano a su límite de 125 °C, se pueda incluir en la variable error y modificar la apertura de las válvulas. De igual manera, en la asignación de los valores para las funciones de pertenencia del error, consideramos a 5 variables lingüísticas: error negativo grande (erng), error negativo pequeño (ernp), error cero (erz), error positivo pequeño (erpp) y error positivo grande (erpg). A estas variables lingüísticas les otorgamos valores que consideramos óptimos para el funcionamiento del sistema, para el error negativo grande se utiliza una función trapezoidal que toma los valores de la figura 172 que son [-70, -60, -2, -1.5]; para el error negativo pequeño también se utiliza una función trapezoidal que toma los valores de la figura 173 que son [-2, -1.5, -0.2, 0]; para el error cero se utiliza una función triangular que toma los valores de la figura 174 que son [-0.025, 0, 0.025]; para el error positivo pequeño, al igual que su homónimo negativo, también se utiliza una función trapezoidal que toma los valores de la figura 175 que son [0, 0.02, 1.5, 2]; de igual manera, para el error positivo grande se utiliza una función trapezoidal que toma los valores de la figura 176 que son [1.5, 2, 60, 70];

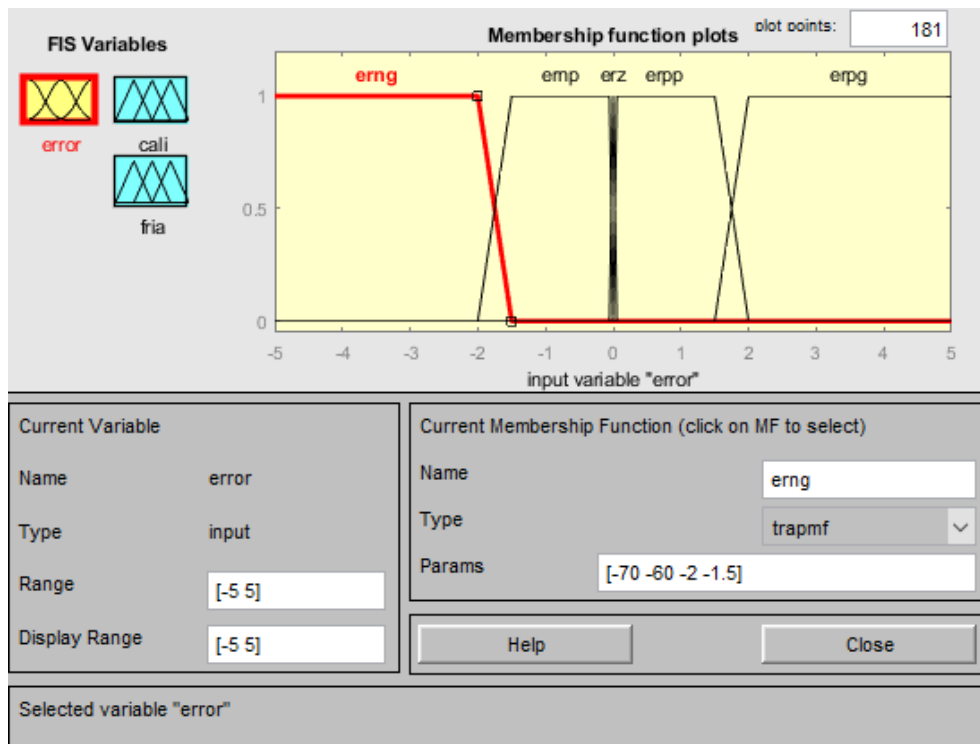


Figura 172 Error negativo grande.

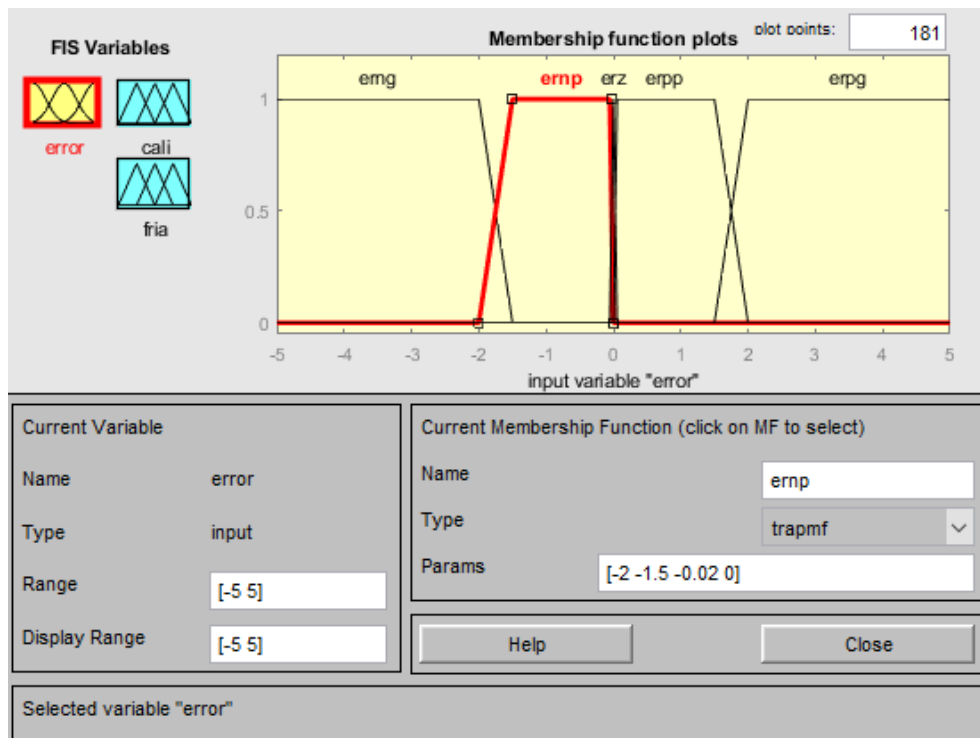


Figura 173 Error negativo pequeño.

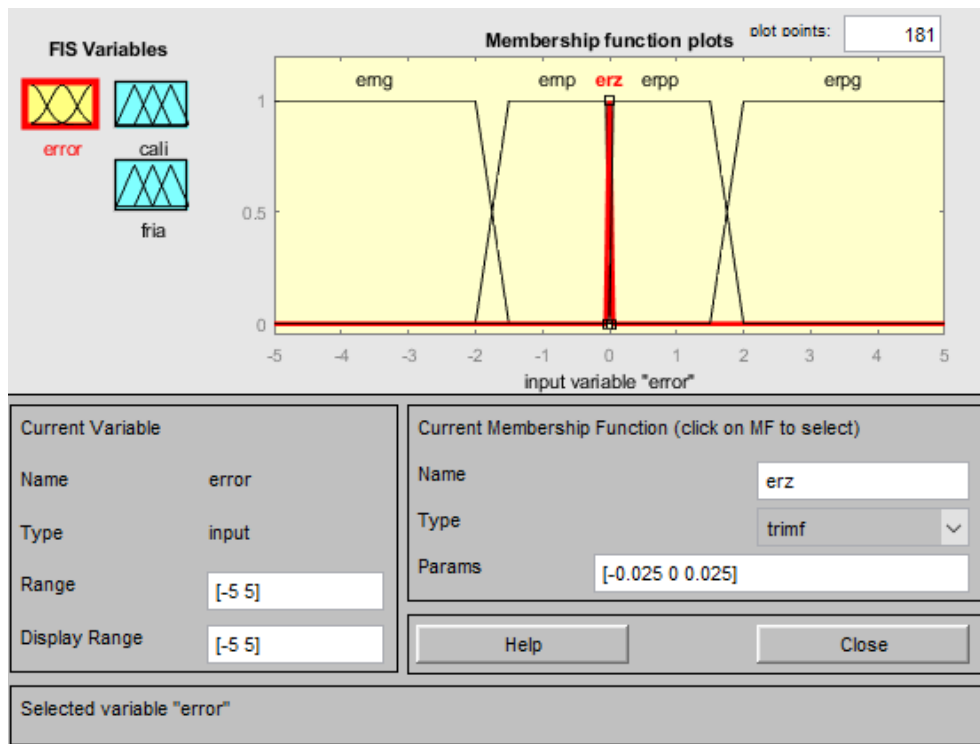


Figura 174 Error cero.

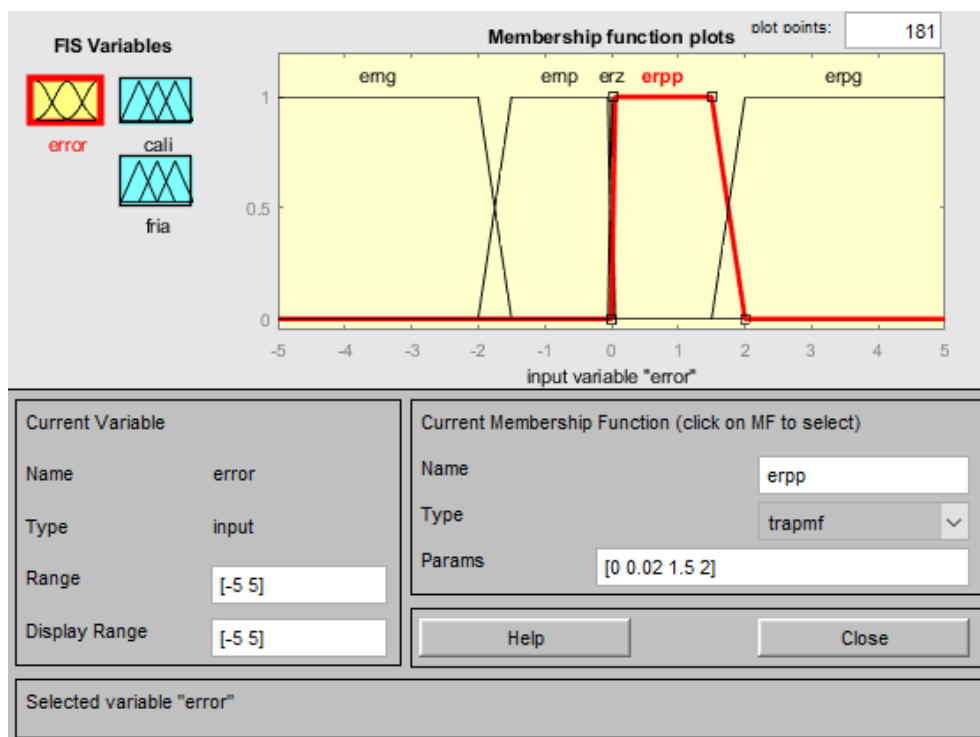
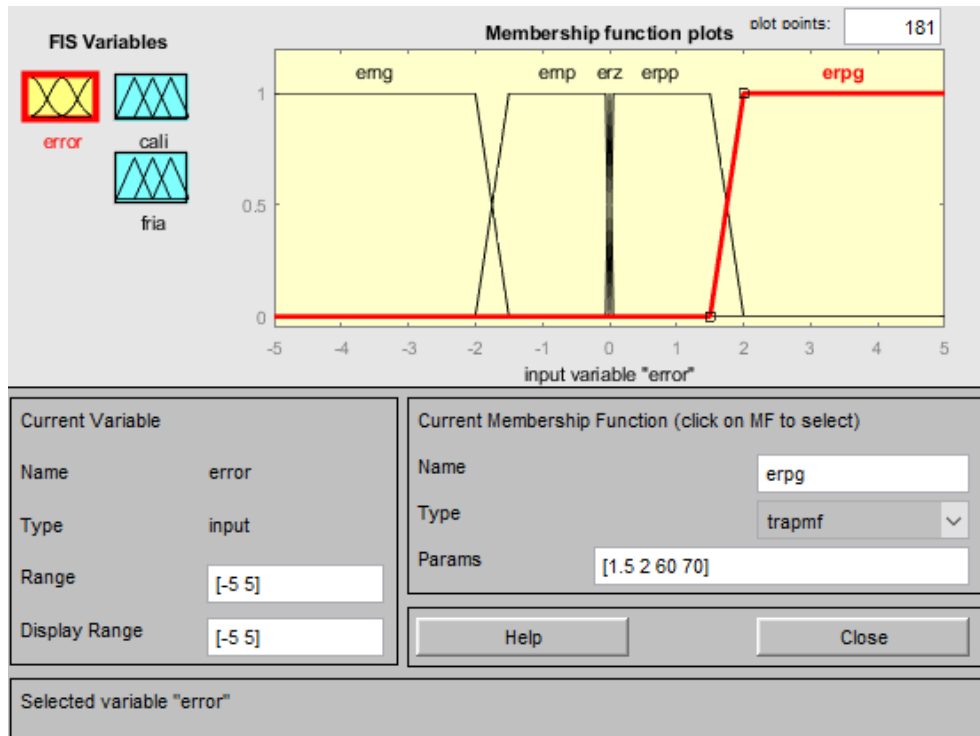


Figura 175 Error positivo pequeño.



*Figura 176 Error positivo grande*

Las otras dos variables “cali” y “fria” representan los pasos que debe dar el motor NEMA 17 para abrir una válvula en su totalidad. La apertura de las válvulas está determinada por la cantidad de pasos que puede dar el motor a pasos NEMA 17, el cual, como ya se ha mencionado, tiene una cantidad de 200 pasos por revolución, los cuales consideramos al inicio del desarrollo de este control difuso, pues las válvulas que conocemos y que están instaladas en nuestras regaderas cumplen con la característica de poder dar una vuelta completa para abrirse por completo, sin embargo, al encontrar la versión de válvulas para regadera que utilizamos en el prototipo como una versión más barata y de un buen material, descubrimos que son de 1/4 de vuelta. Por lo cual optamos por reducir esa cantidad de 200 a solo 50, que es la cantidad de pasos que requiere el motor para dar 1/4 de vuelta. Por lo cual, en la asignación de los valores para las funciones de pertenencia de las variables “cali” y “fría”, consideramos a 5 variables lingüísticas: cerrado (cerr), poco abierto (poco), medio abierto (med), casi abierto (casi) y abierto (abi). Dando como resultado los siguientes valores para las funciones de pertenencia triangulares de ambas variables: para cerrado los valores son de la figura 177 [0, 0, 12.5]; para poco abierto los valores son los de la figura 178 [0, 12.5, 25]; para medio abierto los valores son los de la figura 179 [12.5,

25, 37.5]; para casi abierto los valores son los de la figura 180 [25, 37.5, 50]; y para abierto los valores son los de la figura 181 [37.5, 50, 50].

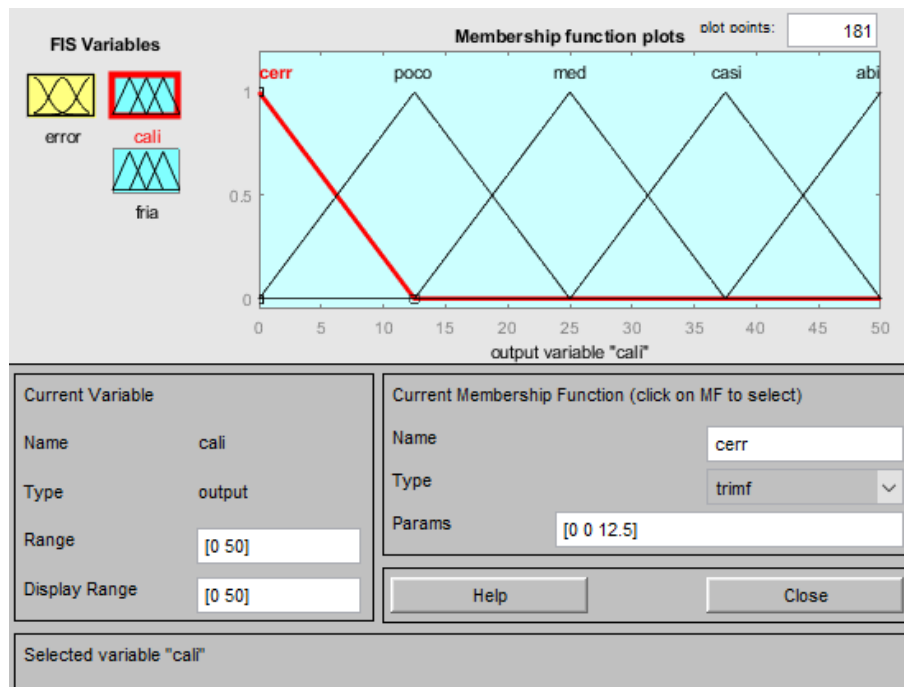


Figura 177 Cerrado

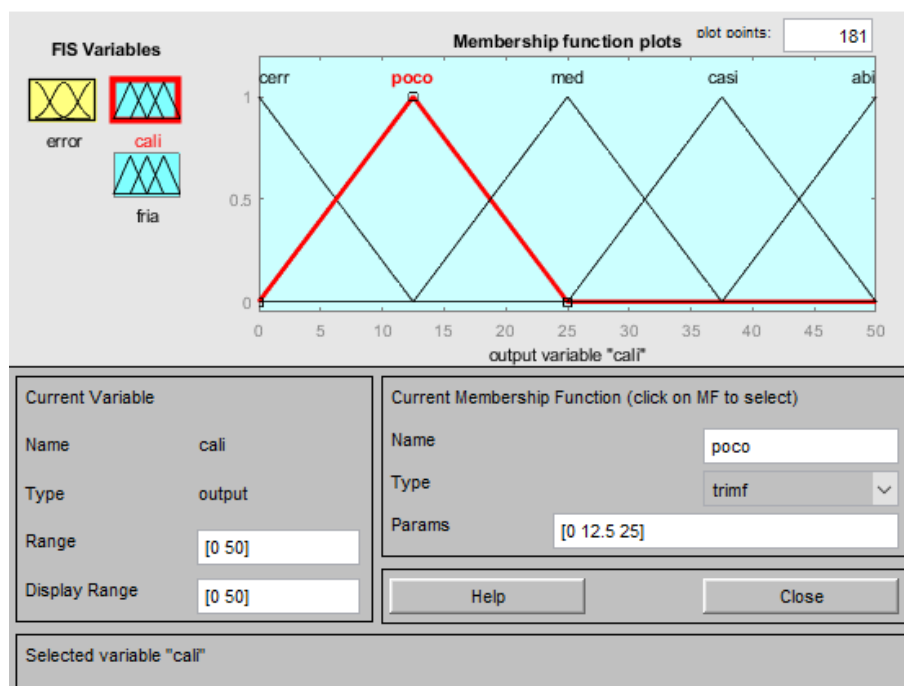


Figura 178 Poco abierto

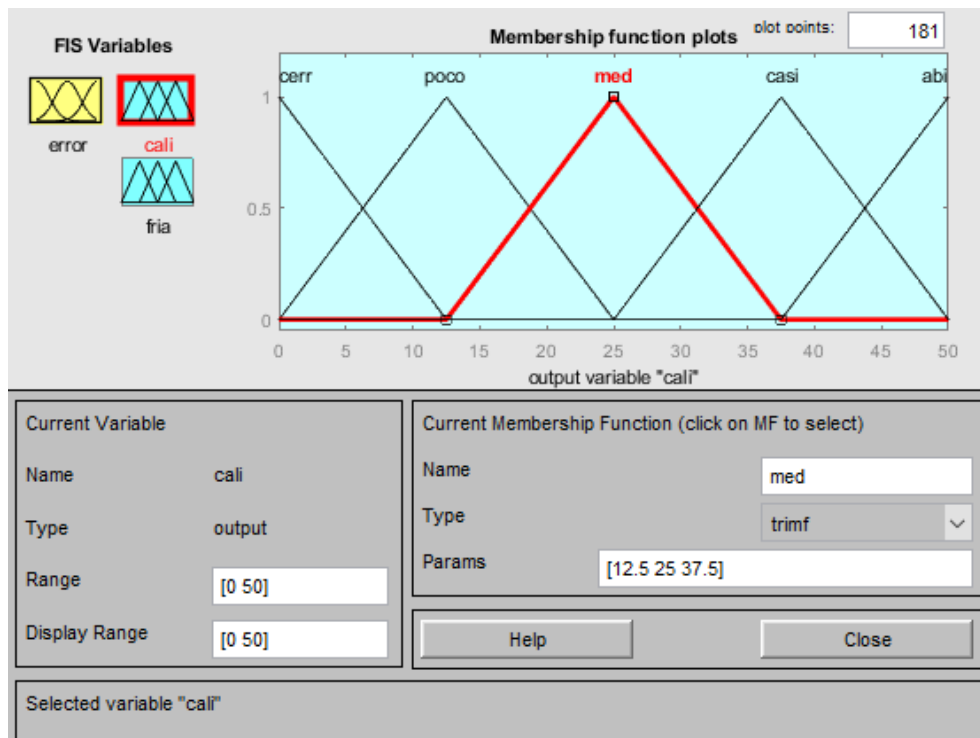


Figura 179 Medio abierto.

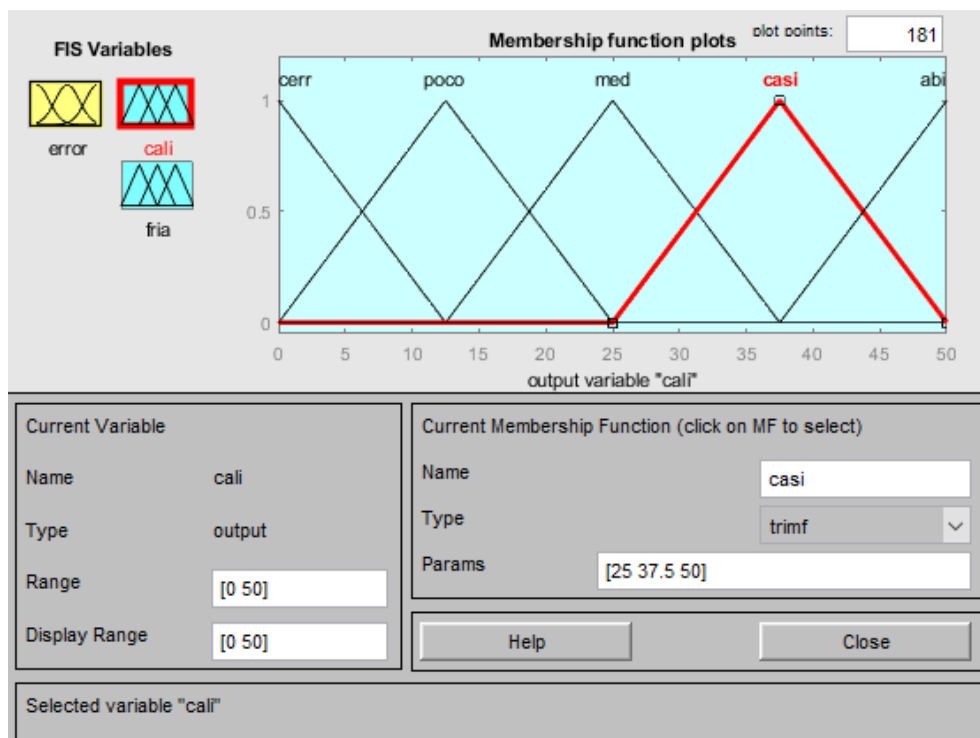
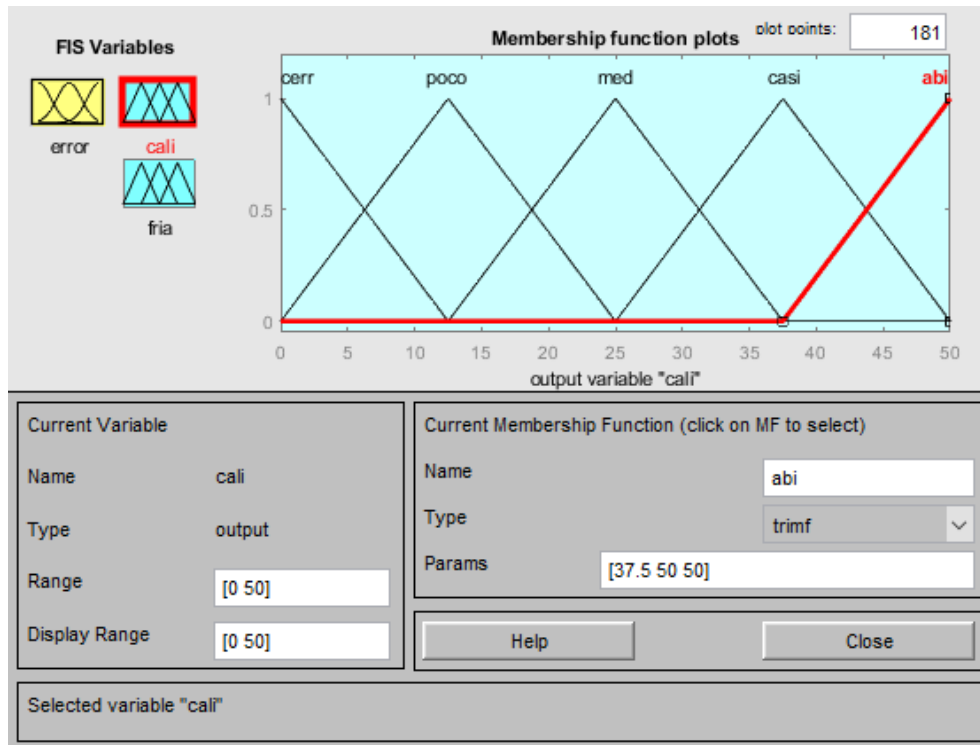


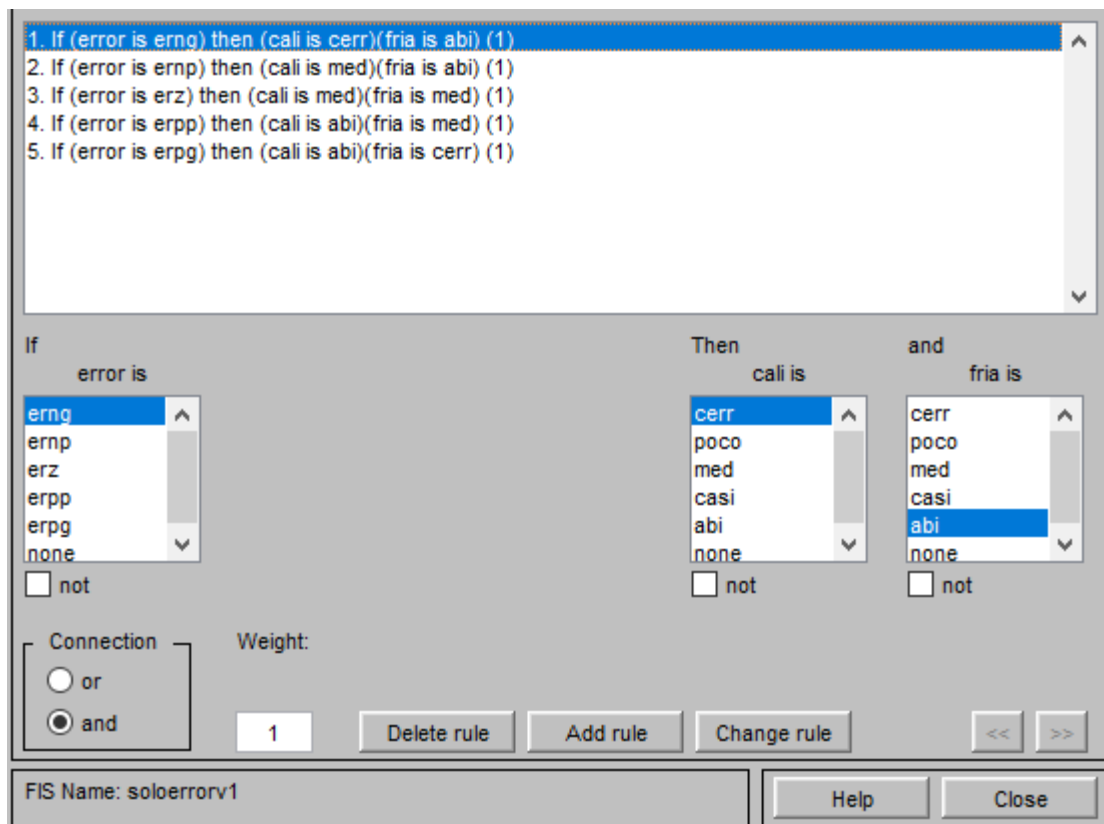
Figura 180 Casi abierto.





*Figura 181 Abierto.*

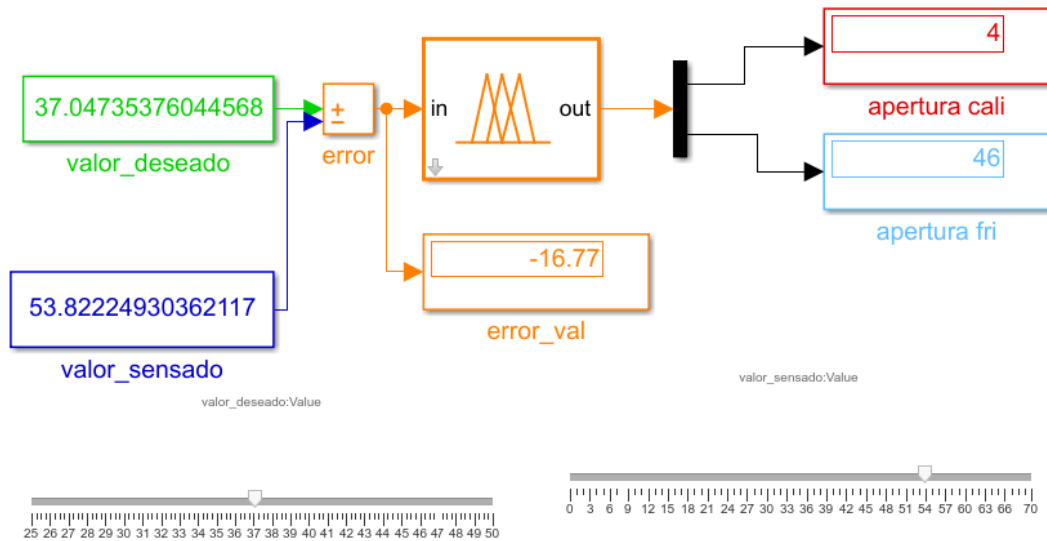
Una vez definidos los valores de las 3 variables, se realizan las reglas de la figura 182 que seguirá el control difuso, las cuales siguen las necesidades de temperatura en el agua mezclada para igualar a la temperatura deseada, puesto que si el error es negativo grande, se indica que la temperatura sensada es mucho mayor que la temperatura deseada, por lo cual la válvula del agua caliente toma un valor entre cerrado y poco abierto, mientras que el agua fría fluye en mayor cantidad, disminuyendo la temperatura del agua mezclada. Ocurre algo similar cuando el error es negativo pequeño, pues la válvula de agua caliente se abre aproximadamente a la mitad y la fría fluye en abundancia. En caso de haber un error cero o muy cercano a cero, las válvulas quedarán abiertas a la mitad, esperando que cualquier variación en la temperatura sensada los haga modificar su valor de acuerdo con las demás reglas. Cuando se tiene un error positivo pequeño, se indica que falta una pequeña cantidad de temperatura para llegar a la temperatura deseada, por lo cual es necesario abrir más la válvula de agua caliente y dejar a la mitad al agua fría. El error positivo grande manifiesta una mayor necesidad de calor que el error positivo pequeño, por lo cual, el agua fría se encontrará cerrada y la caliente abierta.



*Figura 182 Variables de control difuso.*

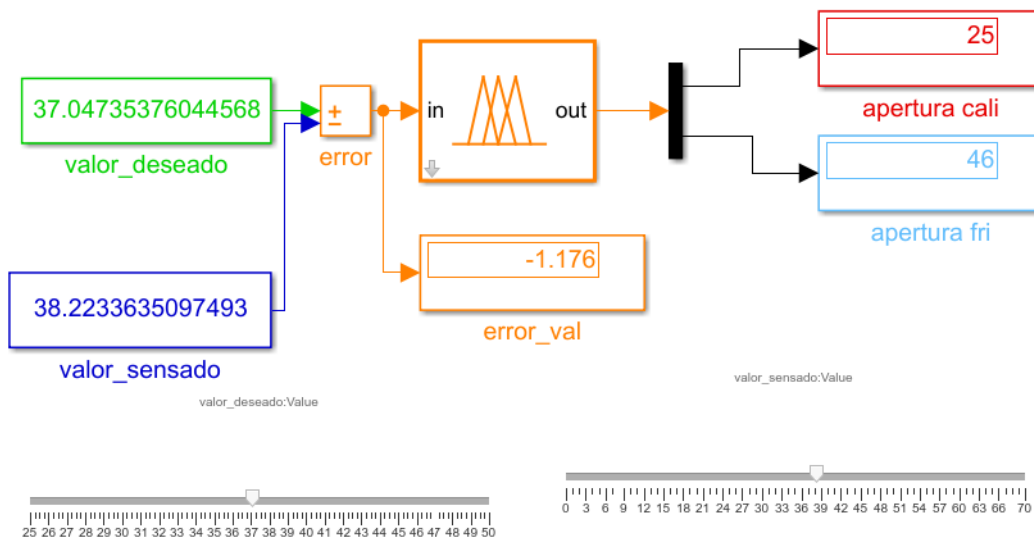
Después del diseño de las reglas, realizamos el esquema de Simulink que mencionamos en el apartado de diseño de control difuso en el software Matlab y su análogo mediante programación en Python y realizamos diversas pruebas, donde modificamos manualmente los valores del “valor\_deseado” y el “valor\_sensado”, obteniendo los siguientes resultados.

Al introducir un error negativo grande de -16.77 como en la figura 183, las válvulas tomaron los siguientes valores de apertura: de 4 para la de agua caliente y de 46 para la de agua fría.



*Figura 183 Error negativo grande.*

Por otra parte, cuando se coloca un error negativo pequeño de -1.176 como en la figura 184, las válvulas tomaron los siguientes valores de apertura: de 25 para la de agua caliente y de 46 para la de agua fría.



*Figura 184 Error negativo pequeño.*

Mientras que con un error muy cercano a cero de -0.006093 como en la figura 185, las válvulas tomaron los siguientes valores de apertura: de 25 para la de agua caliente y de 29.34 para la de agua fría.

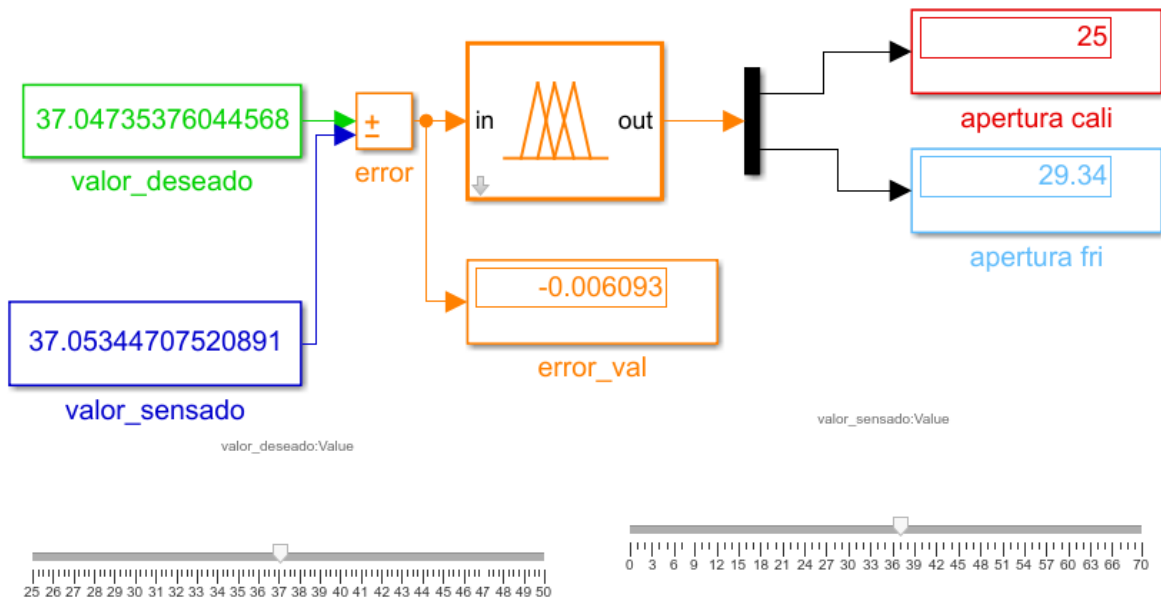


Figura 185 Error cercano a cero.

En el caso de introducir un error positivo pequeño de 1.749 como en la figura 186, las válvulas tomaron los siguientes valores de apertura: de 45.28 para la de agua caliente y de 18.16 para la de agua fría.

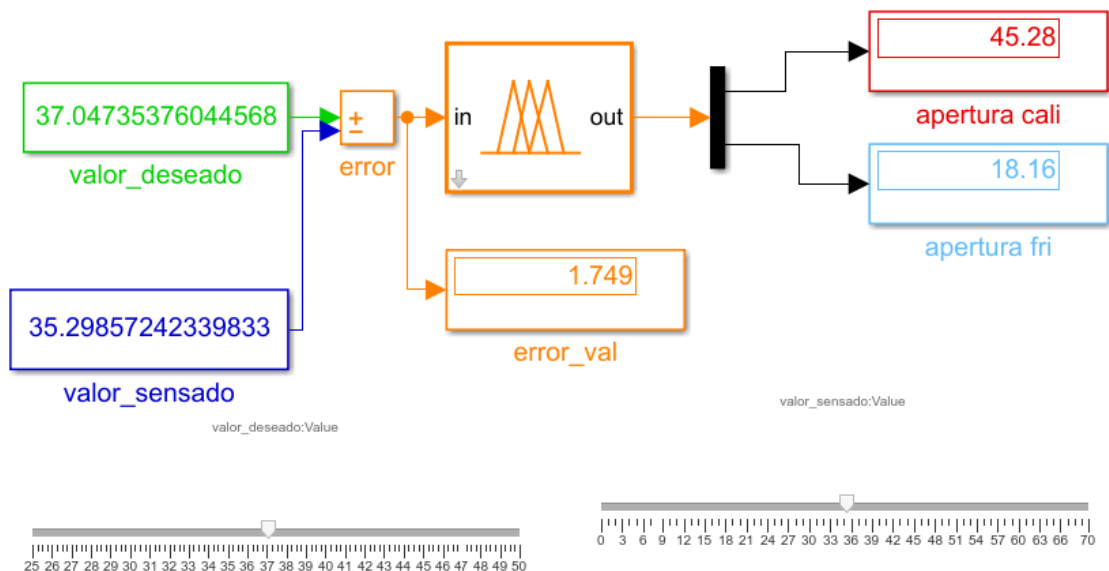
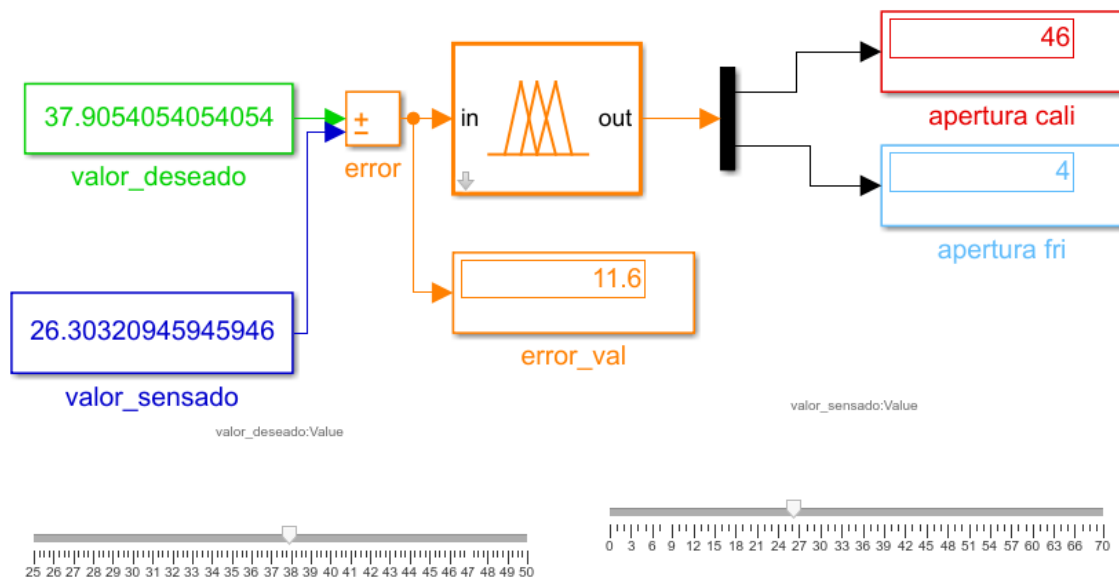


Figura 186 Error positivo pequeño.

Por último, al introducir un error positivo grande de 11.6 como en la figura 187, las válvulas tomaron los siguientes valores de apertura: de 46 para la de agua caliente y de 4 para la de agua fría.

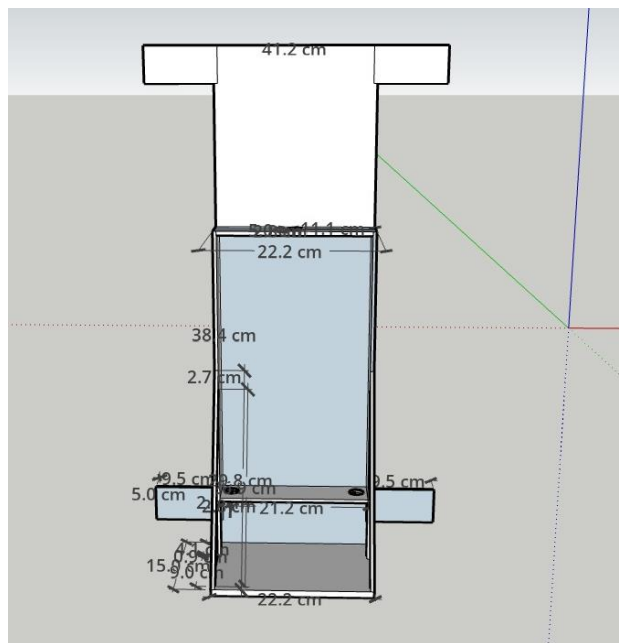


*Figura 187 Error positivo grande.*

De esta manera, la simulación del control difuso mediante Matlab demostró que el control planteado podría cumplir las expectativas del sistema y que con una correcta implementación dentro del código en Python podría regularse automáticamente para que el error sea siempre cercano a cero, con un margen mínimo de diferencia entre el valor de la temperatura deseada y el valor de la temperatura sensada. Las funciones de pertenencia y reglas editadas desde la herramienta de edición de lógica difusa de Matlab son las mismas que se utilizan en el control difuso escrito en Python, pues son reducidas y sencillas de procesar para diversos microcontroladores. Sin embargo, la Raspberry pi 4 se eligió porque el código consta de muchas más acciones (como la recolección de datos de flujo y temperatura, el control de los motores, la creación de una interfaz de usuario en el panel táctil y en aplicación web) y por la facilidad de programar con librerías de Python permite que la codificación más eficiente de dichas acciones.

## 4. 2 Integración de bloques y software.

Para la integración de los bloques y el software, se une el sistema hidráulico – electrónico con el programa que ya ha sido modificado para albergar a todos los códigos creados en lenguaje Python de las pruebas preliminares, el control difuso y las interfaces de usuario. Con tal motivo, es necesario indicar las medidas de nuestro prototipo dentro de su modelo 3D, pues como ya se ha mencionado en el dimensionamiento y la creación del prototipo, las medidas de los componentes del prototipo físico fueron contempladas para desarrollar el modelo 3D. En las figuras 188 a la 191 se muestran diferentes perspectivas del modelo 3D de la estructura de MDF creado en SketchUp.



*Figura 188 Vista frontal del modelo de la estructura de MDF en 3D.*

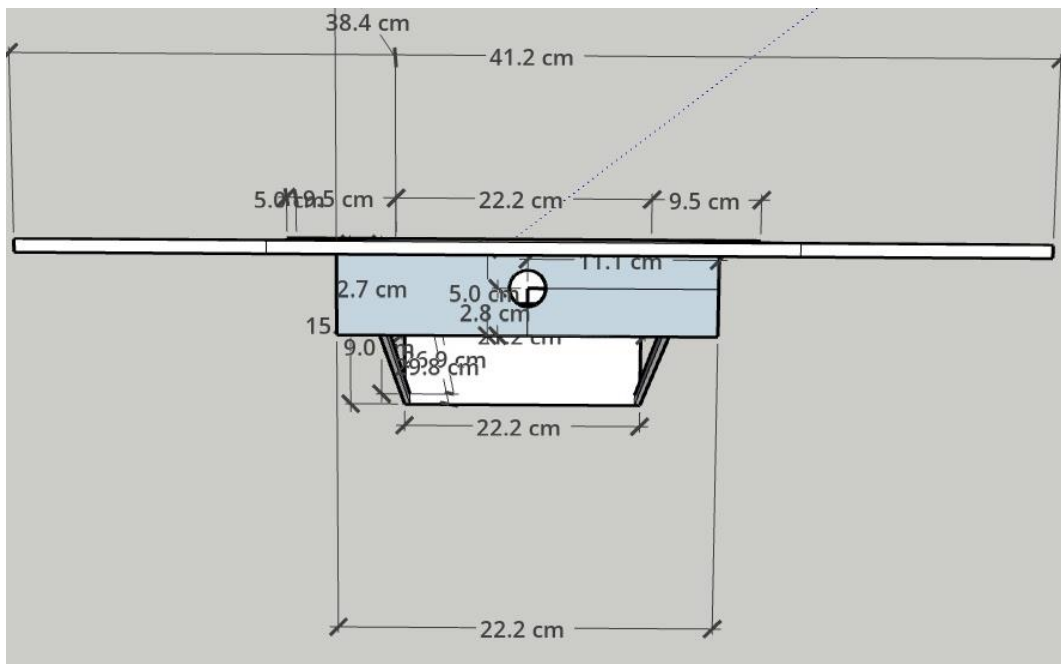


Figura 189 Vista superior del modelo de la estructura de MDF en 3D.

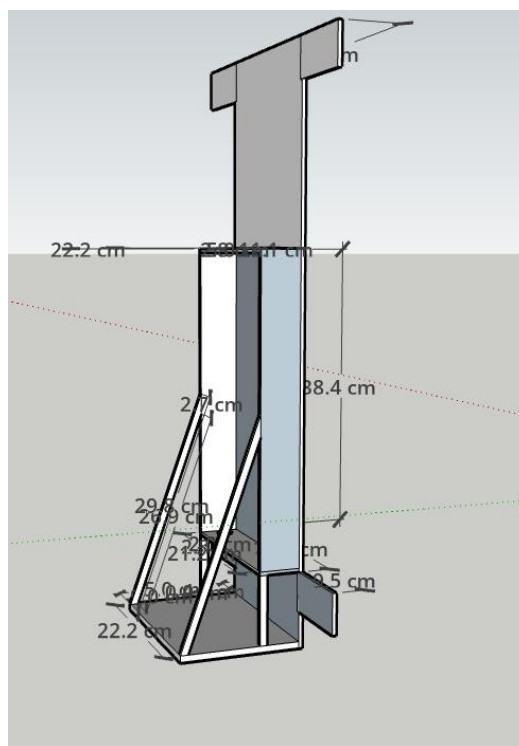
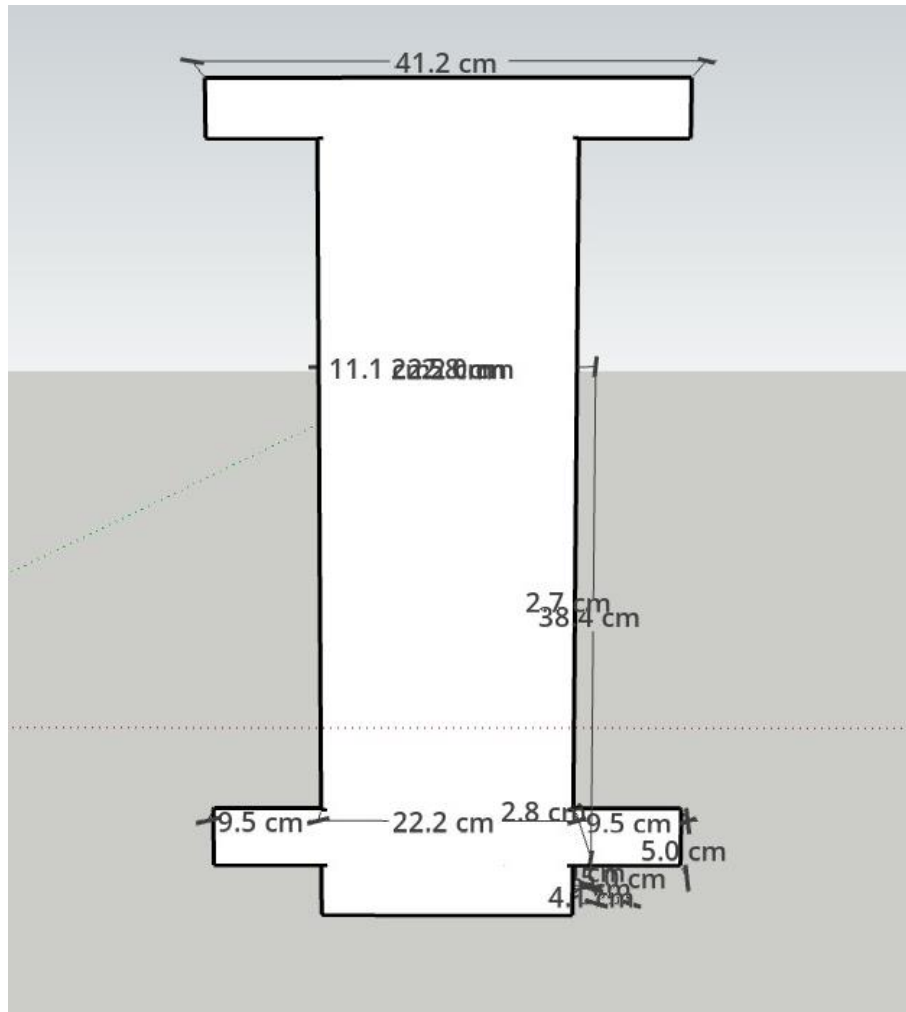


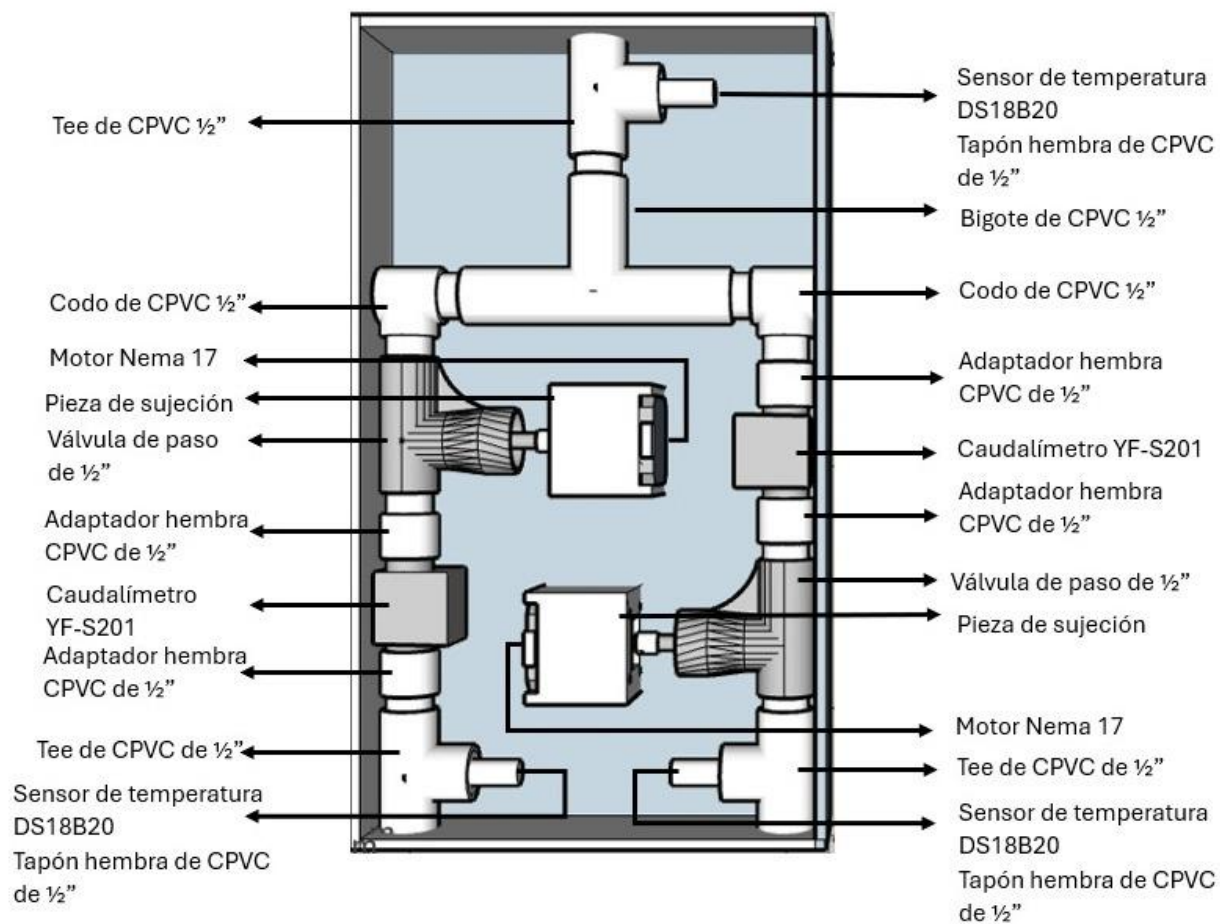
Figura 190 Vista lateral del modelo de la estructura de MDF en 3D.



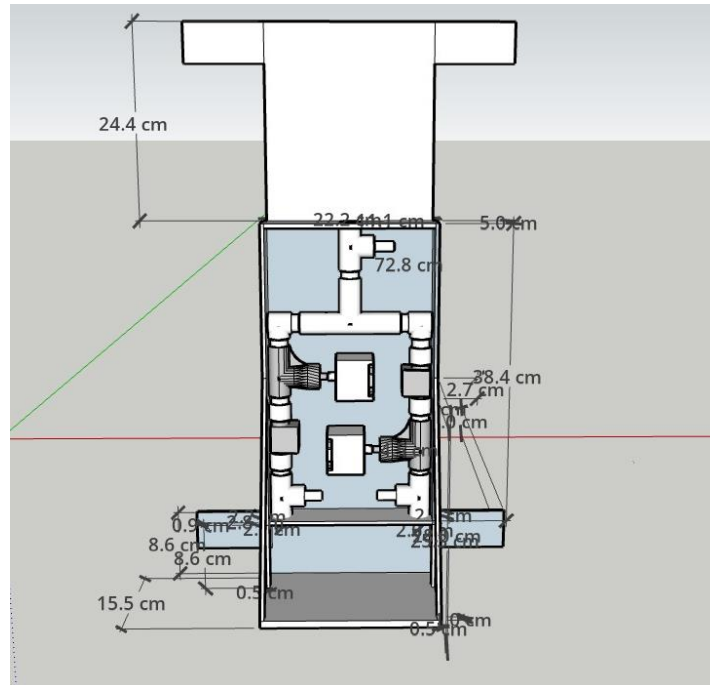


*Figura 191 Vista posterior del modelo de la estructura de MDF en 3D.*

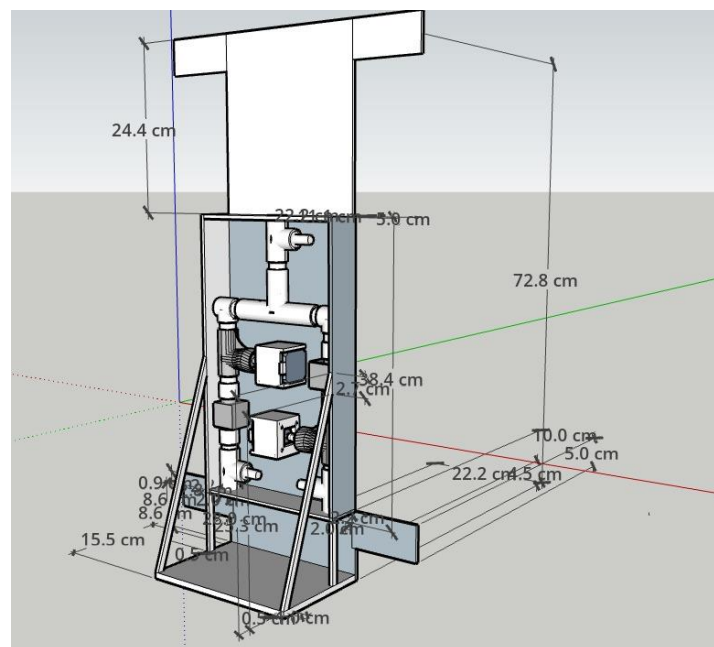
Como se puede observar en el apartado de la creación 3D del prototipo por medio de software, las medidas de la estructura de MDF fueron predispuestas a las medidas reales de cada componente, de tal modo que la caja de la estructura de MDF queda como se muestra en la figura 192, y en las figuras 193 y 194 se muestran diferentes perspectivas del prototipo 3D en general con sus medidas reales.



*Figura 192 Elementos del sistema hidráulico – electrónico en el modelo 3D.*

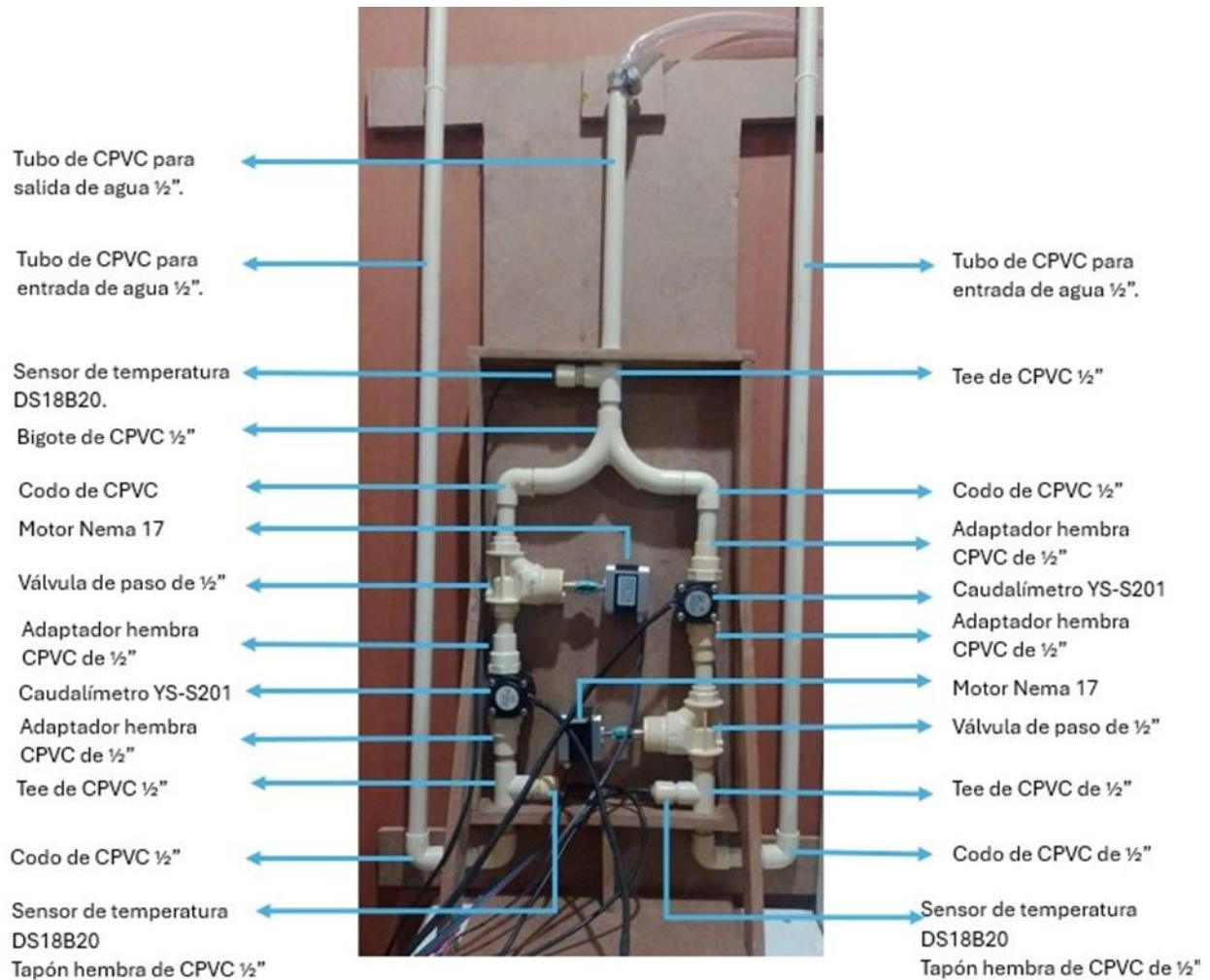


*Figura 193 Vista frontal del prototipo modelado en 3D con medidas.*



*Figura 194 Vista lateral del prototipo modelado en 3D con medidas.*

A partir de este modelo, se crea la integración de los bloques y el software, pues con el sistema ensamblado de la sección de implementación, podemos realizar las conexiones pertinentes de los sensores y actuadores a la Raspberry pi como se observa en las figuras 195, 196 y 197.



*Figura 195 Estructura MDF.*



*Figura 196 Conexiones de elementos electrónicos.*



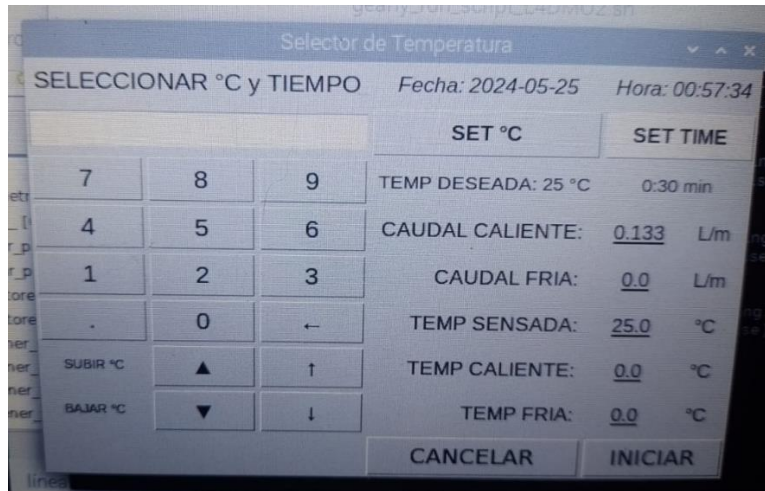
*Figura 197 Conexiones en Raspberry Pi*

### **4.3 Demostración del funcionamiento del sistema completo.**

Una vez que el sistema hidráulico está integrado en su totalidad con el sistema electrónico se procede a encender la Raspberry Pi para comenzar la ejecución final del programa (ver anexo 6), el cual está basado en la programación orientada a objetos debido a la complejidad del sistema. Este programa es capaz de leer la temperatura deseada por el usuario desde la aplicación web o desde el panel táctil, además de mostrar los datos de los sensores utilizados en el prototipo y modificar la posición de los motores según los resultados del control difuso. A continuación, se muestran los resultados obtenidos del funcionamiento del sistema completo.

En las figuras 198 y 199 se muestra el sistema antes de ser iniciado, con una temperatura deseada de 25°C y un tiempo de temporizador de 30 segundos (0:30 min), además se puede apreciar la fecha, la hora y el valor de la temperatura sensada, que es la única necesaria para comprobar el sistema, sin embargo, se tienen los sensores para las tuberías individualmente en caso de ser necesarios para alguna medición en ellas.





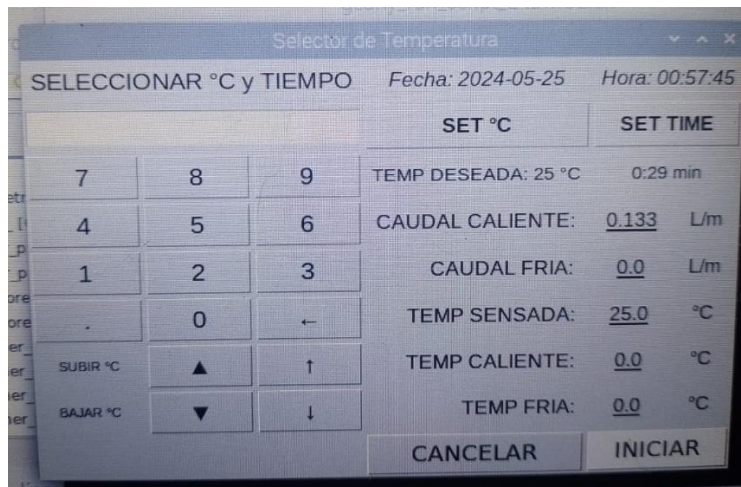
*Figura 198 Inicialización del sistema con duración de 30 seg.*



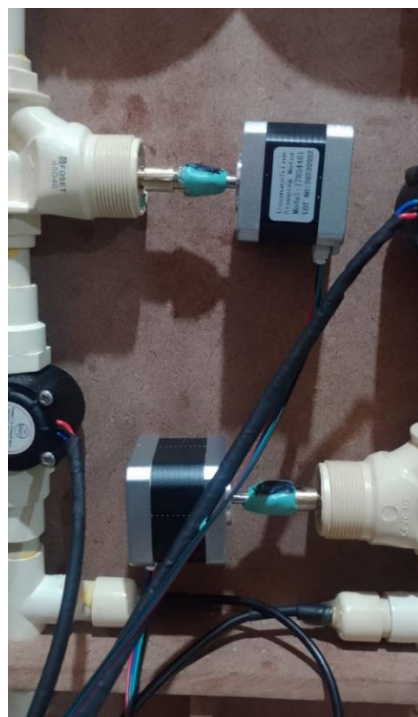
*Figura 199 Válvulas cerradas al inicio de la prueba*



Al momento de presionar el botón “INICIAR”, el sistema comienza el temporizador y los motores comienzan a girar como se aprecia en las figuras 200 y 201. debido a que las temperaturas deseada y sensada coinciden, el error es cero, por lo cual, las válvulas están en una posición de apertura media.

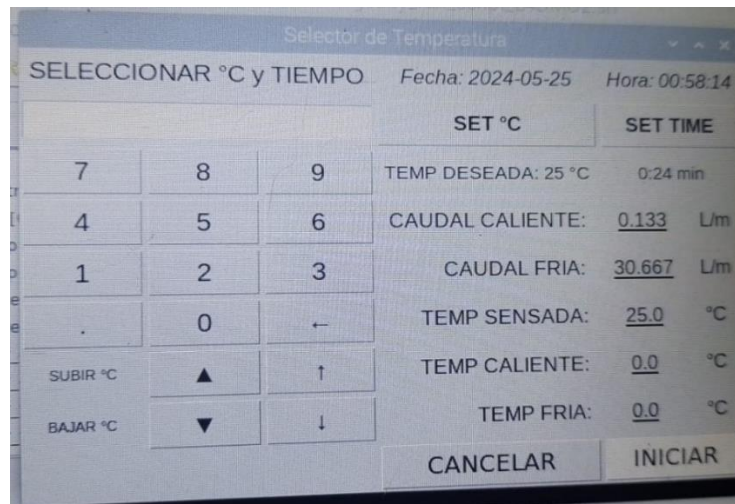


*Figura 200 Sistema iniciado con temperaturas deseada y sensada iguales.*



*Figura 201 Válvulas abiertas a la mitad.*

Conforme avanza el tiempo, el temporizador disminuye y los motores continúan en la posición anterior como se aprecia en las figuras 202 y 203. Debido a que las temperaturas deseada y sensada siguen coincidiendo, el error sigue siendo cero, por lo cual, las válvulas continúan en una posición de apertura media.



*Figura 202 Sistema en funcionamiento con temperaturas deseada y sensada iguales.*



*Figura 203 Válvulas abiertas a la mitad.*

Al faltar 17 segundos, la temperatura sensada varía, pues al estarse enfriando el agua del contenedor de agua caliente (lo dejamos al aire libre para que la temperatura pudiera disminuir según se enfriaba el entorno) cambió también la temperatura de la mezcla, como se muestra en la figura 204, mientras que las válvulas continuaron abiertas a la mitad como se observa en la figura 205.

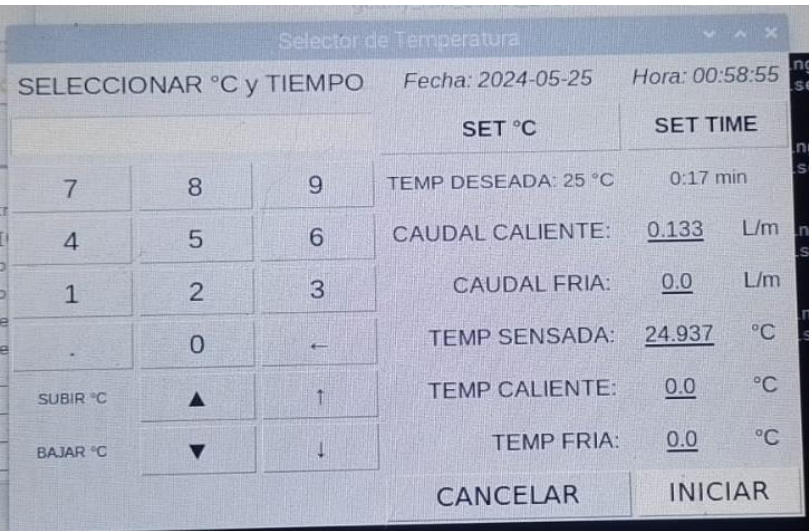


Figura 204 Ligeró cambio en la temperatura sensada.



Figura 205 Válvulas abiertas a la mitad debido a que el error es cercano a cero.

Cuando faltaron 10 segundos para terminar la ducha, se disminuyó el valor de la temperatura del agua caliente a 22°C (figura 206), para observar el cambio en la apertura de las válvulas que está en la figura 207, en la cual, La válvula de agua fría se abrió casi al máximo y la del agua caliente esta casi cerrada.



*Figura 206 Reducción de la temperatura deseada a 22°C.*



*Figura 207 Cambio en la apertura de las válvulas .*



En las figuras 208 y 209 se aprecia que el tiempo del temporizador volvió a cero, por lo cual el sistema reconoce que el tiempo de ducha ha concluido y regresa a las válvulas a su posición inicial de cerrado.



*Figura 208 Temporizador en cero y temperatura deseada igual a la sensada.*



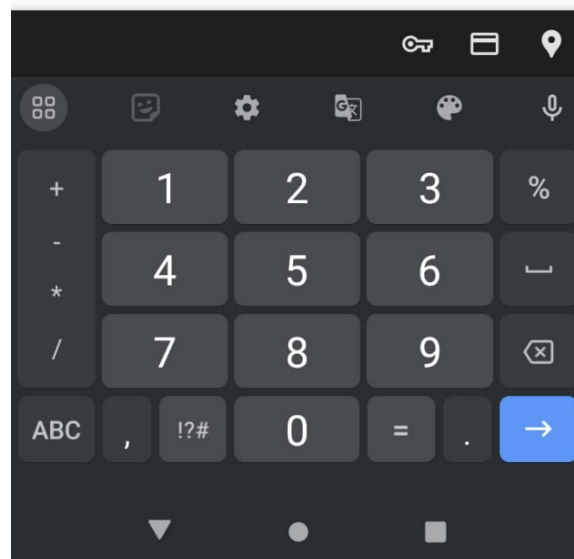
*Figura 209 Válvulas cerradas por completo.*

A partir de estos resultados, realizamos otras pruebas, pero esta vez utilizando la interfaz web que fue desarrollada previamente en lenguaje HTML (ver anexo 7) y de manera similar, se observó su funcionamiento obteniendo los siguientes resultados.

Iniciamos colocando la temperatura y el tiempo de ducha dentro de las secciones mostradas en las figuras 210 y 211



The screenshot shows a mobile web browser interface. At the top, the status bar displays the time 1:46 and various icons. The browser's address bar shows the URL 192.168.1.106:5000. Below the address bar, the page title is "Selector de Temperatura". Underneath, there is a section labeled "Fecha: | Hora:". This section contains two input fields: "Temperatura Deseada:" with the value "0 °C" and "Tiempo (min:seg):" with the value "00:00". Below these fields are two buttons: "Iniciar" and "Cancelar". A blue dot is positioned over the "00:00" text in the time input field. Below the input fields, the section "Estado Actual" displays the following information: "Temperatura Sensada: 0 °C", "Temperatura Caliente: 50 °C", "Temperatura Fría: 21 °C", "Caudal 1: 0 L/m", and "Caudal 2: 0 L/m".



*Figura 210 Introducir tiempo*

1:46

192.168.1.106:5000

### Selector de Temperatura

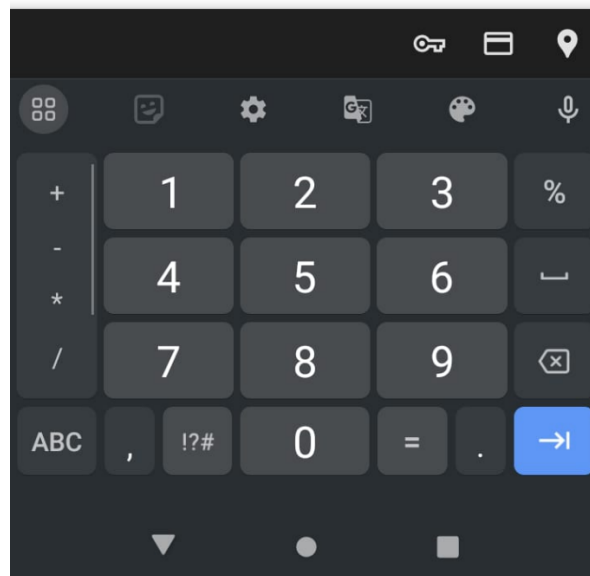
Fecha: | Hora:

Temperatura Deseada: 0 °C Tiempo (min:seg): 00:00

Iniciar Cancelar

### Estado Actual

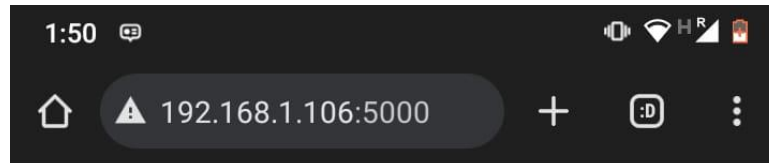
Temperatura Sensada: 0 °C  
Temperatura Caliente: 50 °C  
Temperatura Fría: 21 °C  
Caudal 1: 0 L/m  
Caudal 2: 0 L/m



*Figura 211 Introducir temperatura.*

Posteriormente, al pulsar el botón “iniciar” se observa como la temperatura aumenta gradualmente y al pasar unos segundos, la temperatura sensada es cercana a la temperatura deseada como se muestra en la figura 212.





## Selector de Temperatura

Fecha: | Hora:

| Temperatura Deseada: | Tiempo (min:seg): |
|----------------------|-------------------|
| 36 °C                | 12:21             |
| Iniciar              | Cancelar          |

## Estado Actual

Temperatura Sensada: 34 °C

Temperatura Caliente: 50 °C

Temperatura Fría: 21 °C

Caudal 1: 3 L/m

Caudal 2: 0.133 L/m

*Figura 212 Aplicación web en curso.*

Con unos segundos más de avance, la temperatura puede verse más acercada a la solicitada por el usuario, como en la figura 213.

## Selector de Temperatura

Fecha: | Hora:

| Temperatura Deseada: | Tiempo (min:seg): |
|----------------------|-------------------|
| 36 °C                | 12:19             |
| Iniciar              | Cancelar          |

## Estado Actual

Temperatura Sensada: 34.8 °C

Temperatura Caliente: 50 °C

Temperatura Fría: 21 °C

Caudal 1: 3 L/m

Caudal 2: 0.133 L/m

*Figura 213 Aumento en la temperatura sensada.*

También se realizó la prueba desde una laptop (figura 214) y los valores son iguales a los del teléfono.

The screenshot shows a web browser window with the title 'Selector de Temperatura'. The address bar shows 'No seguro' and the URL '192.168.1.106:5000'. The browser's bookmark bar includes links to Gmail, YouTube, (751) Charlie Rodd..., Maps, WhatsApp, Remove Backgroun..., SketchUp, Speedtest by Ookla..., Venngage | Editor, replit, JupyterLab, and Todos los favoritos.

The web application content is as follows:

### Selector de Temperatura

Fecha: | Hora:

| Temperatura Deseada: | Tiempo (min seg): |
|----------------------|-------------------|
| 36 °C                | 12:21             |

|         |          |
|---------|----------|
| Iniciar | Cancelar |
|---------|----------|

#### Estado Actual

Temperatura Sensada: 34.8 °C  
Temperatura Caliente: 50 °C  
Temperatura Fria: 21 °C  
Caudal 1: 3 L/m  
Caudal 2: 0.133 L/m

The Windows taskbar at the bottom shows the search bar with 'Buscar', several application icons, a system tray with 'ÚLTIMA HORA', and the date/time '09:35 a. m. 25/05/2024'.

*Figura 214 Prueba de la aplicación web en laptop.*

## Capítulo V. Conclusiones

La ducha inteligente con control difuso representa una aplicación innovadora de la tecnología en el ámbito de los sistemas hidráulicos. Esta ducha incorpora una variedad de tecnologías avanzadas para proporcionar una experiencia de baño personalizada y eficiente en el uso del agua. Integrando tecnologías avanzadas como control difuso, sensores de temperatura, motores a pasos y caudalímetros, la ducha inteligente permite un control preciso y automatizado de la temperatura y el flujo de agua. Con la finalidad de una experiencia de ducha más cómoda y personalizada para el usuario.

La inclusión de un panel táctil, una aplicación web programada en HTML y Python facilita la interacción y el control por parte del usuario, permitiéndole ajustar fácilmente la configuración de la ducha según sus preferencias. Esta interfaz usuario-amigable enfatiza un enfoque centrado en el usuario para mejorar la experiencia de baño.

Además, la ducha inteligente demuestra un enfoque consciente de la eficiencia del agua al incorporar caudalímetros para medir la cantidad de agua caliente y fría utilizada. Esto ayuda a los usuarios a controlar y reducir su consumo de agua, contribuyendo así a la conservación de recursos.

Los componentes hidráulicos específicos, como el uso de materiales CPVC, codos, tee, tubos y tapones hembra, aseguran un funcionamiento seguro y eficiente a largo plazo del sistema hidráulico. El sistema de regulación de temperatura innovador utiliza motores tipo NEMA 17 para abrir y cerrar las válvulas de agua fría y caliente, junto con la retroalimentación de los sensores de temperatura. Esto garantiza una regulación precisa de la temperatura del agua.

El funcionamiento de la ducha inteligente, desde la recopilación de datos de los sensores hasta el control de los motores y los caudalímetros, se lleva a cabo mediante los pines GPIO de una Raspberry Pi 4. Este microordenador actúa como el cerebro central del sistema, coordinando todas las operaciones para garantizar un funcionamiento integrado y eficiente de la ducha.

.

## Competencias desarrolladas y/o aplicadas.

- **Desarrollar programas de aplicación con interfaces gráficas para usuario:** dominio de conocimiento de los elementos básicos para la interacción entre el hombre, sistemas y dispositivos electrónicos. Esto implica poder comprender principios fundamentales de diseño de interfaces, sino también tener habilidades para la implementación efectiva de estas interfaces en entornos de desarrollo de software.
- **Diseñar algoritmos computacionales eficientes y efectivos:** Consideración tanto la lógica de programación como la optimización del rendimiento. Esto incluye la capacidad de descomponer problemas complejos en pasos lógicos y traducirlos en código ejecutable. Además, es crucial seleccionar el lenguaje de programación más adecuado para el proyecto en cuestión, aprovechando sus características y capacidades para lograr los objetivos establecidos.
- **Utilizar la computadora de manera competente:** Habilidades avanzadas en el uso de herramientas de programación junto con la capacidad de buscar información de manera efectiva en medios electrónicos para mantenerse actualizado en un campo de constante evolución. Además, la capacidad de generar informes de investigación claros y concisos es crucial para comunicar resultados y hallazgos de manera efectiva.
- **Manejar equipos de medición y destreza para el desarrollo de las interfaces:** Esto por medio de conocimientos sólidos en electrónica y habilidades prácticas para utilizar herramientas y dispositivos de medición, así como la capacidad de interpretar y analizar los datos obtenidos. Además, se requiere atención meticulosa a los detalles y una comprensión profunda de los principios de funcionamiento de los circuitos electrónicos para garantizar el éxito en el desarrollo de interfaces electrónicas.
- **Habilidad para identificar y diagnosticar problemas:** Basándose en mediciones eléctricas es esencial para el mantenimiento y la resolución de problemas en sistemas eléctricos complejos. Esto puede incluir la detección de cortocircuitos, circuitos abiertos, fallas de componentes y problemas de calidad de energía, entre otros.

- **Conocimiento sobre el funcionamiento de microcontroladores:** Esto se refiere tanto interno como externo del microcontrolador. Además de desarrollar programas tanto en lenguaje ensamblador como en lenguaje de alto nivel, aprovechando los recursos del microcontrolador para abordar problemas específicos.
- **Diseño de circuitos digitales y analógicos, utilizando herramientas computacionales:** Esto implica la capacidad de diseñar y crear circuitos electrónicos, tanto digitales como analógicos. Estas herramientas posibilitan el diseñar, simular y verificar el funcionamiento de los circuitos antes de su ejecución física, lo que ayuda a prevenir errores y optimizar el diseño.
- **Desarrollar programas en lenguajes de alto nivel:** Este punto se refiere a la habilidad para escribir código en lenguajes de programación como Python, Java, C++, entre otros. Estos lenguajes son más fáciles de entender y escribir para los humanos, y permiten desarrollar una amplia variedad de aplicaciones y programas.
- **Interpretar diagramas esquemáticos eléctricos y electrónicos:** Esta habilidad implica poder entender y analizar diagramas que representan la conexión y funcionamiento de circuitos eléctricos y electrónicos. Esto es fundamental para el diseño, la reparación y el mantenimiento de sistemas electrónicos.
- **Seleccionar, aplicar, calibrar y operar instrumentos de medición y control para automatizar procesos industriales:** Implica elegir los instrumentos adecuados, ajustar su precisión y operarlos con seguridad. La automatización requiere programación y configuración para optimizar el rendimiento. Estas habilidades son esenciales para mejorar la eficiencia y la productividad en entornos industriales.

## Referencias

- [1]. Matín Castillo, J. C. M. C. [Juan Carlos], & Ayesa Álvarez, R. A. A. [Rubén]. (2023). Instalaciones domóticas - Novedad 2023. Google Books. Recuperado de: [https://books.google.com.mx/books?id=rLXHEAAQBAJ&pg=PA6&hl=es&source=gb\\_s\\_to\\_c\\_r&cad=2#v=onepage&q&f=false](https://books.google.com.mx/books?id=rLXHEAAQBAJ&pg=PA6&hl=es&source=gb_s_to_c_r&cad=2#v=onepage&q&f=false)
- [2]. Huidobro J. M. (2004). "La domótica entra en nuestras casas." Recuperado de: <https://dialnet.unirioja.es/servlet/articulo?codigo=5098587>
- [3]. Muñoz, O. Q. (2019). Internet de las cosas (IoT) (1.a ed., Vol. 1). Ibukku LLC. Recuperado de: <https://books.google.com.mx/books?id=vnnEDwAAQBAJ&printsec=frontcover&hl=es#v=onepage&q&f=false>
- [4]. Kaspersky "¿Qué es la internet de las cosas? definición y explicación." (2024, 18 enero). Recuperado de: <https://www.kaspersky.es/resource-center/definitions/what-is-iot>
- [5]. Qwerty. (2021, 29 noviembre). Dispositivos IOT. características y ejemplos - Hubler. Hubler. Recuperado de: <https://hubler.es/dispositivos-iot-caracteristicas-y-ejemplos/#:~:text=Tipos%20de%20Dispositivos%20IoT%20en%20la%20actualidad%201,Analytics%20y%20Machine%20Learning.%205%20Inteligencia%20Artificial%20%28IA%209>
- [6]. AWS. "¿Qué es IoT? - Explicación del Internet de las cosas" (s. f.). Amazon Web Services, Inc. Recuperado de: <https://aws.amazon.com/es/what-is/iot/>
- [7]. Lenis, A. (2023, 17 mayo). "Qué es la interfaz de usuario, qué tipos existen y ejemplos." HustSop. Recuperado 24 de enero de 2024, de <https://blog.hubspot.es/website/interfaz-usuario>
- [8]. Johnson, D. (2024, 16 marzo). Fuzzy Logic Tutorial: What is, Architecture, Application, Example. Guru99. Recuperado de: <https://www.guru99.com/es/what-is-fuzzy-logic.html>
- [9]. C, S. (2021, 16 abril). Control Fuzzy – Mamdani – Simulink. Control Automático Educación. Recuperado de: [https://controlautomaticoeducacion.com/control-realimentado/control-fuzzy-mamdani-simulink/#Logica\\_Difusa](https://controlautomaticoeducacion.com/control-realimentado/control-fuzzy-mamdani-simulink/#Logica_Difusa)

- [10]. Planas, O. (26 julio 2021). "Sensor de temperatura: descripción, ejemplos de función y tipos." Recuperado de: <https://solar-energia.net/termodinamica/propiedades-termodinamicas/temperatura/sensores-temperatura>
- [11]. Admin. (2021, 9 octubre). DS18B20 con Arduino: Sensor de temperatura digital. Geek Factory. Recuperado de: <https://www.geekfactory.mx/tutoriales-arduino/ds18b20-con-arduino-sensor-de-temperatura-digital/%7D>
- [12]. Salvador, I. M. (2020, 5 mayo). "Caudalímetro con arduino YF-S201". Blog Arduino, LabVIEW y Electrónica. Recuperado de: <https://electronicamade.com/caudalimetro-con-arduino/>
- [13]. Vicherrez, J. V. (2024, 30 marzo). CAUDALIMETRO YF-S201 . Flujómetros y Caudalímetro. Recuperado de: <https://flujometros-caudalimetros.com/caudalimetro/yf-s201/#:~:text=El%20YF-S201%20es%20un%20caudal%C3%ADmetro%20digital%20que%20mide,retroiluminada%20para%20facilitar%20la%20lectura%20en%20ambientes%20oscuros>
- [14]. Perez, N. (2023, 11 agosto). Motor Nema 17: funcionamiento y características principales. Racingmotor. Recuperado de: [https://www.racingmotor.zonagp.com/motor-nema-17-funcionamiento-y-caracteristicas-principales/#google\\_vignette](https://www.racingmotor.zonagp.com/motor-nema-17-funcionamiento-y-caracteristicas-principales/#google_vignette)
- [15]. De Bakker, B. (2024, 29 enero). Tutorial de TB6600 Stepper Motor Driver con Arduino. Makerguides.com. Recuperado de: <https://www.makerguides.com/es/tb6600-stepper-motor-driver-arduino-tutorial/>
- [16]. Dima. (2023, 19 agosto). Modelos de llaves de paso de agua empotradas: todo lo que necesitas saber - Mi Hogar. Mi Hogar. Recuperado de: <https://mihogar.blog/guias/modelos-de-llaves-de-paso-de-agua-empotradas-todo-lo-que-necesitas-saber/#:~:text=Las%20llaves%20de%20paso%20de%20agua%20empotradas%20son,nos%20permiten%20controlar%20y%20regular%20el%20flujo%20del>
- [17]. [17]. Barnes, R. (Director). (2018). The official Raspberry Pi Beginner's Guide: How to Use Your New Computer (P. King, Ed.; 4.a ed., Vol. 1). Gareth Halfacree.



- [18]. Watters, A. (2023, 24 enero). Java: Guía completa sobre uno de los lenguajes de programación más utilizados. Blog Centro de e-Learning. Recuperado de: <https://blog.centrodeelearning.com/java-todo-lo-que-hay-que-saber-sobre-uno-de-los-lenguajes-de-programacion-mas-utilizados/>
- [19]. Moreno, P., & Moreno, P. (2024, 13 febrero). ¿Qué es el lenguaje C++ y para qué sirve? Blog Centro de e-Learning. Recuperado de: <https://blog.centrodeelearning.com/lenguaje-c/>
- [20]. Luna, A. C. (2019). CREACIÓN DE PÁGINAS WEB: HTML 5 (1.a ed., Vol. 1). ICB, S.L. (Interconsulting Bureau S.L.). Recuperado de: [https://books.google.com.mx/books?hl=es&lr=&id=tSBvDwAAQBAJ&oi=fnd&pg=PT9&dq=PROGRAMACI%C3%93N+EN+HTML&ots=tlyyx\\_crC-&sig=OYGb43pxd46blt-pu5ozrPR8-40#v=onepage&q=PROGRAMACI%C3%93N%20EN%20HTML&f=false](https://books.google.com.mx/books?hl=es&lr=&id=tSBvDwAAQBAJ&oi=fnd&pg=PT9&dq=PROGRAMACI%C3%93N+EN+HTML&ots=tlyyx_crC-&sig=OYGb43pxd46blt-pu5ozrPR8-40#v=onepage&q=PROGRAMACI%C3%93N%20EN%20HTML&f=false)
- [21]. C, P. (2023, 16 mayo). ¿Qué es una Pantalla Táctil y Cómo Funciona? Electrónica Online. Recuperado de: <https://electronicaonline.net/telefonos-moviles/android-guia/que-es-una-pantalla-tactil/>
- [22]. Julián, O. J., & Andrés, O. G. C. (2008). Diseño y desarrollo de la ducha inteligente iShower. Repositorio Institucional Universidad Eia. Recuperado de: <https://repository.eia.edu.co/entities/publication/e7eb7170-e176-4c1e-acb0-08e854148bbe>
- [23]. De Catalunya, U. O., Giménez, F., & Judit, C. B. (2018, 15 junio). Rivus: Diseño y conceptualización de una ducha inteligente. Recuperado de: <https://openaccess.uoc.edu/handle/10609/81970>
- [24]. Sistema de ducha termostático, sistema inteligente de chorro. (s. f.). Cuotas Sin Interés. Recuperado de: [https://articulo.mercadolibre.com.mx/MLM-2580584260-sistema-de-ducha-termostatico-sistema-inteligente-de-chorro-JM#position=11&search\\_layout=grid&type=item&tracking\\_id=c695d70b-8efa-42ed-87b6-98227aeaddb9](https://articulo.mercadolibre.com.mx/MLM-2580584260-sistema-de-ducha-termostatico-sistema-inteligente-de-chorro-JM#position=11&search_layout=grid&type=item&tracking_id=c695d70b-8efa-42ed-87b6-98227aeaddb9)
- [25]. Juego de ducha inteligente, cabezal de ducha LED de techo. (s. f.). Cuotas Sin Interés. Recuperado de: <https://articulo.mercadolibre.com.mx/MLM-1914505621-juego-de>

[ducha-inteligente-cabezal-de-ducha-led-de-techo-](#)  
[JM#position=23&search layout=grid&type=item&tracking id=999d6b83-ca43-4fb0-](#)  
[92ab-d1d422c0f6c0.](#)

- [26]. Panel de ducha de acero inoxidable modelo Imperial Solana regadera de lujo termostática color niquel con pantalla LED inteligente marca temperatura y hora / rociador de hidromasaje y regadera de mano. (s. f.). Envío Gratis. Recuperado de: [https://www.mercadolibre.com.mx/panel-de-ducha-de-acero-inoxidable-modelo-imperial-solana-regadera-de-lujo-termostatica-color-niquel-con-pantalla-led-inteligente-marca-temperatura-y-hora-rociador-de-hidromasaje-y-regadera-de-](#)  
[mano/p/MLM24542052?pdp\\_filters=item id:MLM1947069171#is advertising=true&se](#)  
[archVariation=MLM24542052&position=1&search layout=grid&type=pad&tracking id=](#)  
[211b9e31-90fb-40ba-8661-](#)  
[54e549b24c2a&is advertising=true&ad domain=VQCATCORE LST&ad position=1&ad](#)  
[click id=MDE3OWM0OTQtODhmZi00Yzg4LWI2NjctODM1ZmYzYWNmOGU3](#)
- [27]. Sistema ducha digital baño inteligente indicador. (s. f.). Cuotas Sin Interés. Recuperado de: [https://articulo.mercadolibre.com.mx/MLM-1921175329-sistema-ducha-digital-bano-](#)  
[inteligente-indicador-](#)  
[JM#position=17&search layout=grid&type=item&tracking id=e7791ca0-2d2c-4011-](#)  
[9320-a7bd952d68da](#)

## Anexos.

### Anexo 1. Hoja de datos del sensor de temperatura DS18B20.

PRELIMINARY

  
**DALLAS**  
SEMICONDUCTOR

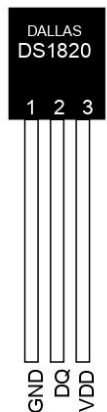
**DS18B20**  
**Programmable Resolution**  
**1-Wire® Digital Thermometer**

www.dalsemi.com

#### FEATURES

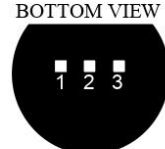
- Unique 1-Wire interface requires only one port pin for communication
- Multidrop capability simplifies distributed temperature sensing applications
- Requires no external components
- Can be powered from data line. Power supply range is 3.0V to 5.5V
- Zero standby power required
- Measures temperatures from -55°C to +125°C. Fahrenheit equivalent is -67°F to +257°F
- $\pm 0.5^\circ\text{C}$  accuracy from -10°C to +85°C
- Thermometer resolution is programmable from 9 to 12 bits
- Converts 12-bit temperature to digital word in 750 ms (max.)
- User-definable, nonvolatile temperature alarm settings
- Alarm search command identifies and addresses devices whose temperature is outside of programmed limits (temperature alarm condition)
- Applications include thermostatic controls, industrial systems, consumer products, thermometers, or any thermally sensitive system

#### PIN ASSIGNMENT



GND  
DQ  
VDD

BOTTOM VIEW



DS18B20 To-92 Package

|                 |   |   |     |
|-----------------|---|---|-----|
| NC              | 1 | 8 | NC  |
| NC              | 2 | 7 | NC  |
| V <sub>DD</sub> | 3 | 6 | NC  |
| DQ              | 4 | 5 | GND |

DS18B20Z  
8-Pin SOIC (150 mil)

#### PIN DESCRIPTION

GND - Ground  
DQ - Data In/Out  
V<sub>DD</sub> - Power Supply Voltage  
NC - No Connect

#### DESCRIPTION

The DS18B20 Digital Thermometer provides 9 to 12-bit (configurable) temperature readings which indicate the temperature of the device.

Information is sent to/from the DS18B20 over a 1-Wire interface, so that only one wire (and ground) needs to be connected from a central microprocessor to a DS18B20. Power for reading, writing, and performing temperature conversions can be derived from the data line itself with no need for an external power source.

Because each DS18B20 contains a unique silicon serial number, multiple DS18B20s can exist on the same 1-Wire bus. This allows for placing temperature sensors in many different places. Applications where this feature is useful include HVAC environmental controls, sensing temperatures inside buildings, equipment or machinery, and process monitoring and control.

**DETAILED PIN DESCRIPTION Table 1**

| PIN<br>8PIN SOIC | PIN<br>TO92 | SYMBOL          | DESCRIPTION                                                                                                                                                        |
|------------------|-------------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 5                | 1           | GND             | Ground.                                                                                                                                                            |
| 4                | 2           | DQ              | <b>Data Input/Output pin.</b> For 1-Wire operation: Open drain. (See “Parasite Power” section.)                                                                    |
| 3                | 3           | V <sub>DD</sub> | <b>Optional V<sub>DD</sub> pin.</b> See “Parasite Power” section for details of connection. V <sub>DD</sub> must be grounded for operation in parasite power mode. |

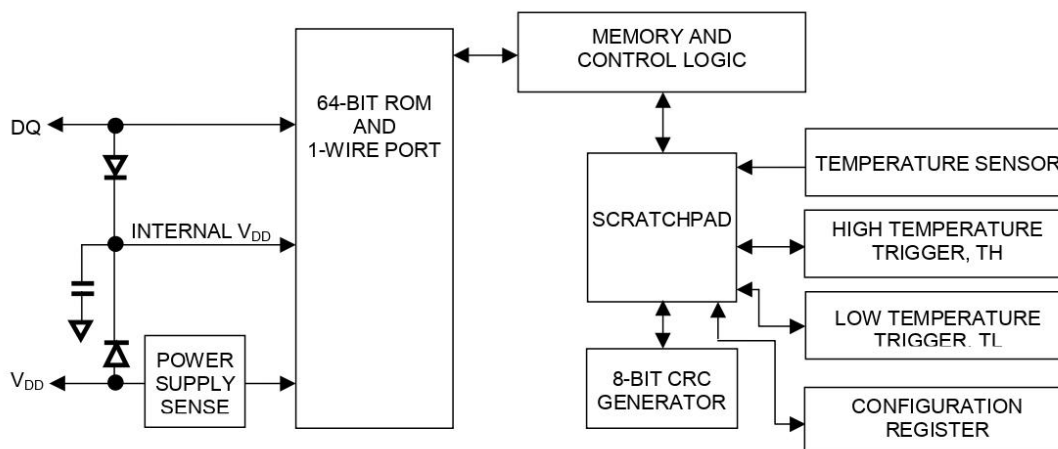
DS18B20Z (8-pin SOIC): All pins not specified in this table are not to be connected.

## OVERVIEW

The block diagram of Figure 1 shows the major components of the DS18B20. The DS18B20 has four main data components: 1) 64-bit lasered ROM, 2) temperature sensor, 3) nonvolatile temperature alarm triggers TH and TL, and 4) a configuration register. The device derives its power from the 1-Wire communication line by storing energy on an internal capacitor during periods of time when the signal line is high and continues to operate off this power source during the low times of the 1-Wire line until it returns high to replenish the parasite (capacitor) supply. As an alternative, the DS18B20 may also be powered from an external 3 volt - 5.5 volt supply.

Communication to the DS18B20 is via a 1-Wire port. With the 1-Wire port, the memory and control functions will not be available before the ROM function protocol has been established. The master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. These commands operate on the 64-bit lasered ROM portion of each device and can single out a specific device if many are present on the 1-Wire line as well as indicate to the bus master how many and what types of devices are present. After a ROM function sequence has been successfully executed, the memory and control functions are accessible and the master may then provide any one of the six memory and control function commands.

One control function command instructs the DS18B20 to perform a temperature measurement. The result of this measurement will be placed in the DS18B20's scratch-pad memory, and may be read by issuing a memory function command which reads the contents of the scratchpad memory. The temperature alarm triggers TH and TL consist of 1 byte EEPROM each. If the alarm search command is not applied to the DS18B20, these registers may be used as general purpose user memory. The scratchpad also contains a configuration byte to set the desired resolution of the temperature to digital conversion. Writing TH, TL, and the configuration byte is done using a memory function command. Read access to these registers is through the scratchpad. All data is read and written least significant bit first.

**DS18B20 BLOCK DIAGRAM** Figure 1

### PARASITE POWER

The block diagram (Figure 1) shows the parasite-powered circuitry. This circuitry “steals” power whenever the DQ or V<sub>DD</sub> pins are high. DQ will provide sufficient power as long as the specified timing and voltage requirements are met (see the section titled “1-Wire Bus System”). The advantages of parasite power are twofold: 1) by parasiting off this pin, no local power source is needed for remote sensing of temperature, and 2) the ROM may be read in absence of normal power.

In order for the DS18B20 to be able to perform accurate temperature conversions, sufficient power must be provided over the DQ line when a temperature conversion is taking place. Since the operating current of the DS18B20 is up to 1.5 mA, the DQ line will not have sufficient drive due to the 5k pullup resistor. This problem is particularly acute if several DS18B20s are on the same DQ and attempting to convert simultaneously.

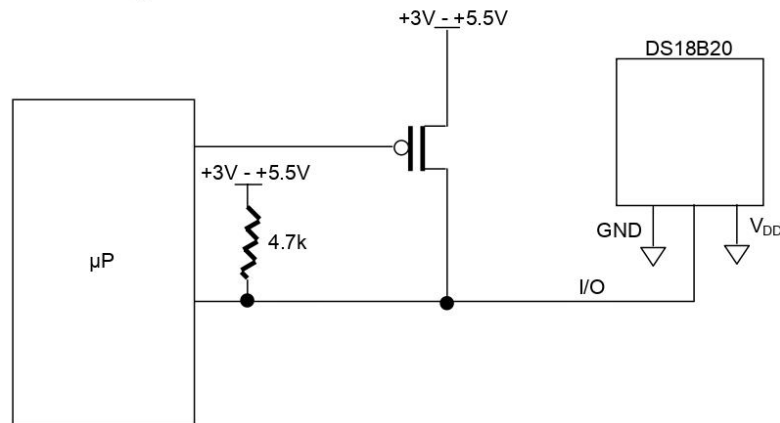
There are two ways to assure that the DS18B20 has sufficient supply current during its active conversion cycle. The first is to provide a strong pullup on the DQ line whenever temperature conversions or copies to the E<sup>2</sup> memory are taking place. This may be accomplished by using a MOSFET to pull the DQ line directly to the power supply as shown in Figure 2. The DQ line must be switched over to the strong pull-up within 10  $\mu$ s maximum after issuing any protocol that involves copying to the E<sup>2</sup> memory or initiates temperature conversions. When using the parasite power mode, the V<sub>DD</sub> pin must be tied to ground.

Another method of supplying current to the DS18B20 is through the use of an external power supply tied to the V<sub>DD</sub> pin, as shown in Figure 3. The advantage to this is that the strong pullup is not required on the DQ line, and the bus master need not be tied up holding that line high during temperature conversions. This allows other data traffic on the 1-Wire bus during the conversion time. In addition, any number of DS18B20s may be placed on the 1-Wire bus, and if they all use external power, they may all simultaneously perform temperature conversions by issuing the Skip ROM command and then issuing the Convert T command. Note that as long as the external power supply is active, the GND pin may not be floating.

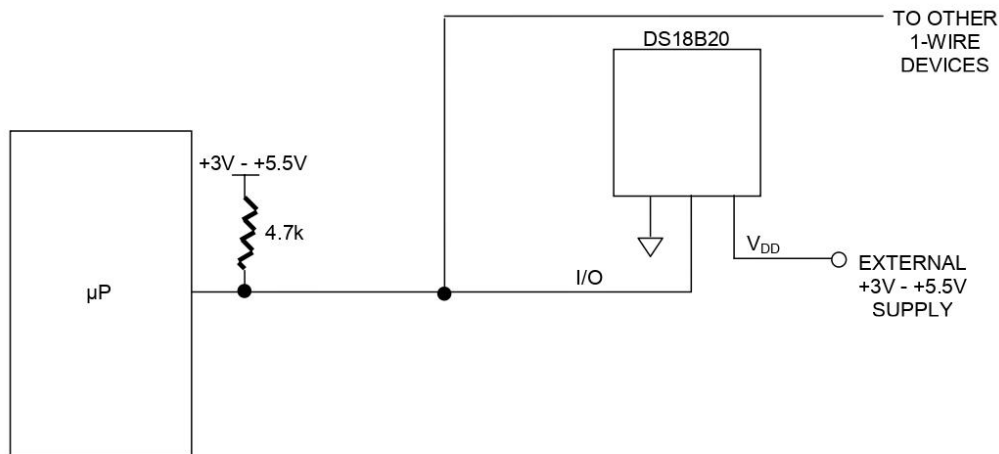
The use of parasite power is not recommended above 100°C, since it may not be able to sustain communications given the higher leakage currents the DS18B20 exhibits at these temperatures. For applications in which such temperatures are likely, it is strongly recommended that V<sub>DD</sub> be applied to the DS18B20.

For situations where the bus master does not know whether the DS18B20s on the bus are parasite powered or supplied with external  $V_{DD}$ , a provision is made in the DS18B20 to signal the power supply scheme used. The bus master can determine if any DS18B20s are on the bus which require the strong pullup by sending a Skip ROM protocol, then issuing the read power supply command. After this command is issued, the master then issues read time slots. The DS18B20 will send back “0” on the 1-Wire bus if it is parasite powered; it will send back a “1” if it is powered from the  $V_{DD}$  pin. If the master receives a “0,” it knows that it must supply the strong pullup on the DQ line during temperature conversions. See “Memory Command Functions” section for more detail on this command protocol.

### STRONG PULLUP FOR SUPPLYING DS18B20 DURING TEMPERATURE CONVERSION Figure 2



### USING $V_{DD}$ TO SUPPLY TEMPERATURE CONVERSION CURRENT Figure 3



## OPERATION - MEASURING TEMPERATURE

The core functionality of the DS18B20 is its direct-to-digital temperature sensor. The resolution of the DS18B20 is configurable (9, 10, 11, or 12 bits), with 12-bit readings the factory default state. This equates to a temperature resolution of 0.5°C, 0.25°C, 0.125°C, or 0.0625°C. Following the issuance of the Convert T [44h] command, a temperature conversion is performed and the thermal data is stored in the scratchpad memory in a 16-bit, sign-extended two's complement format. The temperature information can be retrieved over the 1-Wire interface by issuing a Read Scratchpad [BEh] command once the conversion has been performed. The data is transferred over the 1-Wire bus, LSB first. The MSB of the temperature register contains the “sign” (S) bit, denoting whether the temperature is positive or negative.

Table 2 describes the exact relationship of output data to measured temperature. The table assumes 12-bit resolution. If the DS18B20 is configured for a lower resolution, insignificant bits will contain zeros. For Fahrenheit usage, a lookup table or conversion routine must be used.

**Temperature/Data Relationships Table 2**

|       |       |       |             |          |          |          |          |     |
|-------|-------|-------|-------------|----------|----------|----------|----------|-----|
| $2^3$ | $2^2$ | $2^1$ | $2^0$       | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ | LSB |
| MSb   |       |       | (unit = °C) |          |          |          | LSb      |     |
| S     | S     | S     | S           | S        | $2^6$    | $2^5$    | $2^4$    | MSB |

| TEMPERATURE | DIGITAL OUTPUT<br>(Binary) | DIGITAL<br>OUTPUT<br>(Hex) |
|-------------|----------------------------|----------------------------|
| +125°C      | 0000 0111 1101 0000        | 07D0h                      |
| +85°C       | 0000 0101 0101 0000        | 0550h*                     |
| +25.0625°C  | 0000 0001 1001 0001        | 0191h                      |
| +10.125°C   | 0000 0000 1010 0010        | 00A2h                      |
| +0.5°C      | 0000 0000 0000 1000        | 0008h                      |
| 0°C         | 0000 0000 0000 0000        | 0000h                      |
| -0.5°C      | 1111 1111 1111 1000        | FFF8h                      |
| -10.125°C   | 1111 1111 0101 1110        | FF5Eh                      |
| -25.0625°C  | 1111 1110 0110 1111        | FF6Fh                      |
| -55°C       | 1111 1100 1001 0000        | FC90h                      |

\*The power on reset register value is +85°C.

## OPERATION - ALARM SIGNALING

After the DS18B20 has performed a temperature conversion, the temperature value is compared to the trigger values stored in TH and TL. Since these registers are 8-bit only, bits 9-12 are ignored for comparison. The most significant bit of TH or TL directly corresponds to the sign bit of the 16-bit temperature register. If the result of a temperature measurement is higher than TH or lower than TL, an alarm flag inside the device is set. This flag is updated with every temperature measurement. As long as the alarm flag is set, the DS18B20 will respond to the alarm search command. This allows many DS18B20s to be connected in parallel doing simultaneous temperature measurements. If somewhere the temperature exceeds the limits, the alarming device(s) can be identified and read immediately without having to read non-alarming devices.



## 64-BIT LASERED ROM

Each DS18B20 contains a unique ROM code that is 64-bits long. The first 8 bits are a 1-Wire family code (DS18B20 code is 28h). The next 48 bits are a unique serial number. The last 8 bits are a CRC of the first 56 bits. (See Figure 4.) The 64-bit ROM and ROM Function Control section allow the DS18B20 to operate as a 1-Wire device and follow the 1-Wire protocol detailed in the section “1-Wire Bus System.” The functions required to control sections of the DS18B20 are not accessible until the ROM function protocol has been satisfied. This protocol is described in the ROM function protocol flowchart (Figure 5). The 1-Wire bus master must first provide one of five ROM function commands: 1) Read ROM, 2) Match ROM, 3) Search ROM, 4) Skip ROM, or 5) Alarm Search. After a ROM function sequence has been successfully executed, the functions specific to the DS18B20 are accessible and the bus master may then provide one of the six memory and control function commands.

## CRC GENERATION

The DS18B20 has an 8-bit CRC stored in the most significant byte of the 64-bit ROM. The bus master can compute a CRC value from the first 56-bits of the 64-bit ROM and compare it to the value stored within the DS18B20 to determine if the ROM data has been received error-free by the bus master. The equivalent polynomial function of this CRC is:

$$\text{CRC} = X^8 + X^5 + X^4 + 1$$

The DS18B20 also generates an 8-bit CRC value using the same polynomial function shown above and provides this value to the bus master to validate the transfer of data bytes. In each case where a CRC is used for data transfer validation, the bus master must calculate a CRC value using the polynomial function given above and compare the calculated value to either the 8-bit CRC value stored in the 64-bit ROM portion of the DS18B20 (for ROM reads) or the 8-bit CRC value computed within the DS18B20 (which is read as a ninth byte when the scratchpad is read). The comparison of CRC values and decision to continue with an operation are determined entirely by the bus master. There is no circuitry inside the DS18B20 that prevents a command sequence from proceeding if the CRC stored in or calculated by the DS18B20 does not match the value generated by the bus master.

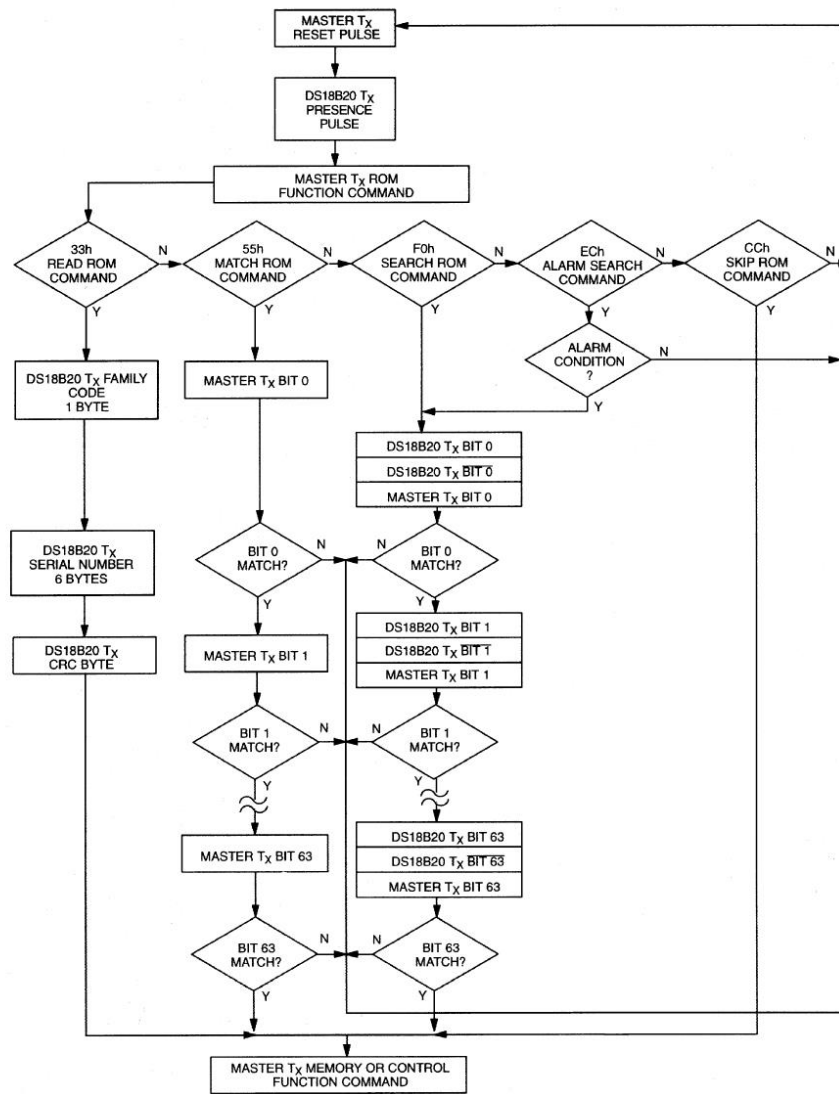
The 1-Wire CRC can be generated using a polynomial generator consisting of a shift register and XOR gates as shown in Figure 6. Additional information about the Dallas 1-Wire Cyclic Redundancy Check is available in Application Note 27 entitled “Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Products.”

The shift register bits are initialized to 0. Then starting with the least significant bit of the family code, 1 bit at a time is shifted in. After the 8th bit of the family code has been entered, then the serial number is entered. After the 48<sup>th</sup> bit of the serial number has been entered, the shift register contains the CRC value. Shifting in the 8 bits of CRC should return the shift register to all 0s.

## 64-BIT LASERED ROM Figure 4

| 8-BIT CRC CODE |     | 48-BIT SERIAL NUMBER |     | 8-BIT FAMILY CODE<br>(28h) |     |
|----------------|-----|----------------------|-----|----------------------------|-----|
| MSB            | LSB | MSB                  | LSB | MSB                        | LSB |

ROM FUNCTIONS FLOW CHART Figure 5





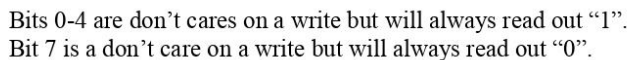
The scratchpad is organized as eight bytes of memory. The first 2 bytes contain the LSB and the MSB of the measured temperature information, respectively. The third and fourth bytes are volatile copies of TH and TL and are refreshed with every power-on reset. The fifth byte is a volatile copy of the configuration register and is refreshed with every power-on reset. The configuration register will be explained in more detail later in this section of the datasheet. The sixth, seventh, and eighth bytes are used for internal computations, and thus will not read out any predictable pattern.

There is a ninth byte which may be read with a Read Scratchpad [BEh] command. This byte contains a cyclic redundancy check (CRC) byte which is the CRC over all of the eight previous bytes. This CRC is implemented in the fashion described in the section titled “CRC Generation”.

## Configuration Register

It contains information which will be used by the device to determine the resolution of the temperature to digital conversion. The bits are organized as shown in Figure 7.

### DS18B20 CONFIGURATION REGISTER Figure 7

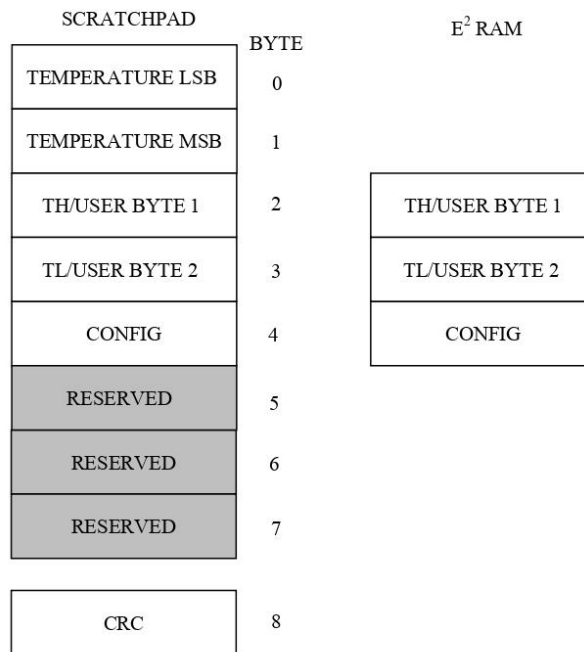


**R0, R1:** Thermometer resolution bits. Table 3 below defines the resolution of the digital thermometer, based on the settings of these 2 bits. There is a direct tradeoff between resolution and conversion time, as depicted in the AC Electrical Characteristics. The factory default of these EEPROM bits is R0=1 and R1=1 (12-bit conversions).

**Thermometer Resolution Configuration Table 3**

| R1 | R0 | Thermometer Resolution | Max Conversion Time              |
|----|----|------------------------|----------------------------------|
| 0  | 0  | 9 bit                  | 93.75 ms ( $t_{\text{conv}}/8$ ) |
| 0  | 1  | 10 bit                 | 187.5 ms ( $t_{\text{conv}}/4$ ) |
| 1  | 0  | 11 bit                 | 375 ms ( $t_{\text{conv}}/2$ )   |
| 1  | 1  | 12 bit                 | 750 ms ( $t_{\text{conv}}$ )     |

**DS18B20 MEMORY MAP Figure 8**



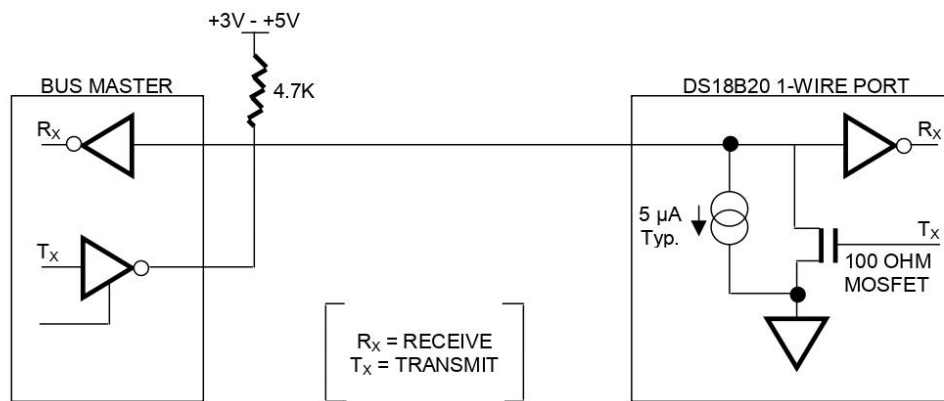
## 1-WIRE BUS SYSTEM

The 1-Wire bus is a system which has a single bus master and one or more slaves. The DS18B20 behaves as a slave. The discussion of this bus system is broken down into three topics: hardware configuration, transaction sequence, and 1-Wire signaling (signal types and timing).

### HARDWARE CONFIGURATION

The 1-Wire bus has only a single line by definition; it is important that each device on the bus be able to drive it at the appropriate time. To facilitate this, each device attached to the 1-Wire bus must have open drain or 3-state outputs. The 1-Wire port of the DS18B20 (DQ pin) is open drain with an internal circuit equivalent to that shown in Figure 9. A multidrop bus consists of a 1-Wire bus with multiple slaves attached. The 1-Wire bus requires a pullup resistor of approximately 5 k $\Omega$ .

### HARDWARE CONFIGURATION Figure 9



The idle state for the 1-Wire bus is high. If for any reason a transaction needs to be suspended, the bus **MUST** be left in the idle state if the transaction is to resume. Infinite recovery time can occur between bits so long as the 1-Wire bus is in the inactive (high) state during the recovery period. If this does not occur and the bus is left low for more than 480  $\mu s$ , all components on the bus will be reset.

### TRANSACTION SEQUENCE

The protocol for accessing the DS18B20 via the 1-Wire port is as follows:

- Initialization
- ROM Function Command
- Memory Function Command
- Transaction/Data

## INITIALIZATION

All transactions on the 1-Wire bus begin with an initialization sequence. The initialization sequence consists of a reset pulse transmitted by the bus master followed by presence pulse(s) transmitted by the slave(s).

The presence pulse lets the bus master know that the DS18B20 is on the bus and is ready to operate. For more details, see the “1-Wire Signaling” section.

## ROM FUNCTION COMMANDS

Once the bus master has detected a presence, it can issue one of the five ROM function commands. All ROM function commands are 8 bits long. A list of these commands follows (refer to flowchart in Figure 5):

### Read ROM [33h]

This command allows the bus master to read the DS18B20's 8-bit family code, unique 48-bit serial number, and 8-bit CRC. This command can only be used if there is a single DS18B20 on the bus. If more than one slave is present on the bus, a data collision will occur when all slaves try to transmit at the same time (open drain will produce a wired AND result).

### Match ROM [55h]

The match ROM command, followed by a 64-bit ROM sequence, allows the bus master to address a specific DS18B20 on a multidrop bus. Only the DS18B20 that exactly matches the 64-bit ROM sequence will respond to the following memory function command. All slaves that do not match the 64-bit ROM sequence will wait for a reset pulse. This command can be used with a single or multiple devices on the bus.

### Skip ROM [CCh]

This command can save time in a single drop bus system by allowing the bus master to access the memory functions without providing the 64-bit ROM code. If more than one slave is present on the bus and a Read command is issued following the Skip ROM command, data collision will occur on the bus as multiple slaves transmit simultaneously (open drain pull downs will produce a wired AND result).

### Search ROM [F0h]

When a system is initially brought up, the bus master might not know the number of devices on the 1-Wire bus or their 64-bit ROM codes. The search ROM command allows the bus master to use a process of elimination to identify the 64-bit ROM codes of all slave devices on the bus.

### Alarm Search [ECh]

The flowchart of this command is identical to the Search ROM command. However, the DS18B20 will respond to this command only if an alarm condition has been encountered at the last temperature measurement. An alarm condition is defined as a temperature higher than TH or lower than TL. The alarm condition remains set as long as the DS18B20 is powered up, or until another temperature measurement reveals a non-alarming value. For alarming, the trigger values stored in EEPROM are taken into account. If an alarm condition exists and the TH or TL settings are changed, another temperature conversion should be done to validate any alarm conditions.

### Example of a ROM Search

The ROM search process is the repetition of a simple three-step routine: read a bit, read the complement of the bit, then write the desired value of that bit. The bus master performs this simple, three-step routine on each bit of the ROM. After one complete pass, the bus master knows the contents of the ROM in one device. The remaining number of devices and their ROM codes may be identified by additional passes.

The following example of the ROM search process assumes four different devices are connected to the same 1-Wire bus. The ROM data of the four devices is as shown:

|      |             |
|------|-------------|
| ROM1 | 00110101... |
| ROM2 | 10101010... |
| ROM3 | 11110101... |
| ROM4 | 00010001... |

The search process is as follows:

1. The bus master begins the initialization sequence by issuing a reset pulse. The slave devices respond by issuing simultaneous presence pulses.
2. The bus master will then issue the Search ROM command on the 1-Wire bus.
3. The bus master reads a bit from the 1-Wire bus. Each device will respond by placing the value of the first bit of their respective ROM data onto the 1-Wire bus. ROM1 and ROM4 will place a 0 onto the 1-Wire bus, i.e., pull it low. ROM2 and ROM3 will place a 1 onto the 1-Wire bus by allowing the line to stay high. The result is the logical AND of all devices on the line, therefore the bus master sees a 0. The bus master reads another bit. Since the Search ROM data command is being executed, all of the devices on the 1-Wire bus respond to this second read by placing the complement of the first bit of their respective ROM data onto the 1-Wire bus. ROM1 and ROM4 will place a 1 onto the 1-Wire, allowing the line to stay high. ROM2 and ROM3 will place a 0 onto the 1-Wire, thus it will be pulled low. The bus master again observes a 0 for the complement of the first ROM data bit. The bus master has determined that there are some devices on the 1-Wire bus that have a 0 in the first position and others that have a 1.

The data obtained from the two reads of the three-step routine have the following interpretations:

|    |                                                                                |
|----|--------------------------------------------------------------------------------|
| 00 | There are still devices attached which have conflicting bits in this position. |
| 01 | All devices still coupled have a 0-bit in this bit position.                   |
| 10 | All devices still coupled have a 1-bit in this bit position.                   |
| 11 | There are no devices attached to the 1-Wire bus.                               |

4. The bus master writes a 0. This deselects ROM2 and ROM3 for the remainder of this search pass, leaving only ROM1 and ROM4 connected to the 1-Wire bus.
5. The bus master performs two more reads and receives a 0-bit followed by a 1-bit. This indicates that all devices still coupled to the bus have 0s as their second ROM data bit.
6. The bus master then writes a 0 to keep both ROM1 and ROM4 coupled.
7. The bus master executes two reads and receives two 0-bits. This indicates that both 1-bits and 0-bits exist as the 3rd bit of the ROM data of the attached devices.



8. The bus master writes a 0-bit. This deselects ROM1, leaving ROM4 as the only device still connected.
9. The bus master reads the remainder of the ROM bits for ROM4 and continues to access the part if desired. This completes the first pass and uniquely identifies one part on the 1-Wire bus.
10. The bus master starts a new ROM search sequence by repeating steps 1 through 7.
11. The bus master writes a 1-bit. This decouples ROM4, leaving only ROM1 still coupled.
12. The bus master reads the remainder of the ROM bits for ROM1 and communicates to the underlying logic if desired. This completes the second ROM search pass, in which another of the ROMs was found.
13. The bus master starts a new ROM search by repeating steps 1 through 3.
14. The bus master writes a 1-bit. This deselects ROM1 and ROM4 for the remainder of this search pass, leaving only ROM2 and ROM3 coupled to the system.
15. The bus master executes two Read time slots and receives two 0s.
16. The bus master writes a 0-bit. This decouples ROM3 leaving only ROM2.
17. The bus master reads the remainder of the ROM bits for ROM2 and communicates to the underlying logic if desired. This completes the third ROM search pass, in which another of the ROMs was found.
18. The bus master starts a new ROM search by repeating steps 13 through 15.
19. The bus master writes a 1-bit. This decouples ROM2, leaving only ROM3.
20. The bus master reads the remainder of the ROM bits for ROM3 and communicates to the underlying logic if desired. This completes the fourth ROM search pass, in which another of the ROMs was found.

**NOTE:**

The bus master learns the unique ID number (ROM data pattern) of one 1-Wire device on each ROM Search operation. The time required to derive the part's unique ROM code is:

$$960 \mu\text{s} + (8 + 3 \times 64) 61 \mu\text{s} = 13.16 \text{ ms}$$

The bus master is therefore capable of identifying 75 different 1-Wire devices per second.

**I/O SIGNALING**

The DS18B20 requires strict protocols to insure data integrity. The protocol consists of several types of signaling on one line: reset pulse, presence pulse, write 0, write 1, read 0, and read 1. All of these signals, with the exception of the presence pulse, are initiated by the bus master.

The initialization sequence required to begin any communication with the DS18B20 is shown in Figure 11. A reset pulse followed by a presence pulse indicates the DS18B20 is ready to send or receive data given the correct ROM command and memory function command.

The bus master transmits (TX) a reset pulse (a low signal for a minimum of 480  $\mu$ s). The bus master then releases the line and goes into a receive mode (RX). The 1-Wire bus is pulled to a high state via the 5k pullup resistor. After detecting the rising edge on the DQ pin, the DS18B20 waits 15-60  $\mu$ s and then transmits the presence pulse (a low signal for 60-240  $\mu$ s).

## MEMORY COMMAND FUNCTIONS

The following command protocols are summarized in Table 4, and by the flowchart of Figure 10.

### Write Scratchpad [4Eh]

This command writes to the scratchpad of the DS18B20, starting at the TH register. The next 3 bytes written will be saved in scratchpad memory at address locations 2 through 4. All 3 bytes must be written before a reset is issued.

### Read Scratchpad [BEh]

This command reads the contents of the scratchpad. Reading will commence at byte 0 and will continue through the scratchpad until the ninth (byte 8, CRC) byte is read. If not all locations are to be read, the master may issue a reset to terminate reading at any time.

### Copy Scratchpad [48h]

This command copies the scratchpad into the E<sup>2</sup> memory of the DS18B20, storing the temperature trigger bytes in nonvolatile memory. If the bus master issues read time slots following this command, the DS18B20 will output 0 on the bus as long as it is busy copying the scratchpad to E<sup>2</sup>; it will return a 1 when the copy process is complete. If parasite-powered, the bus master has to enable a strong pullup for at least 10 ms immediately after issuing this command. The DS18B20 EEPROM is rated for a minimum of 50,000 writes and 10 years data retention at T=+55°C.

### Convert T [44h]

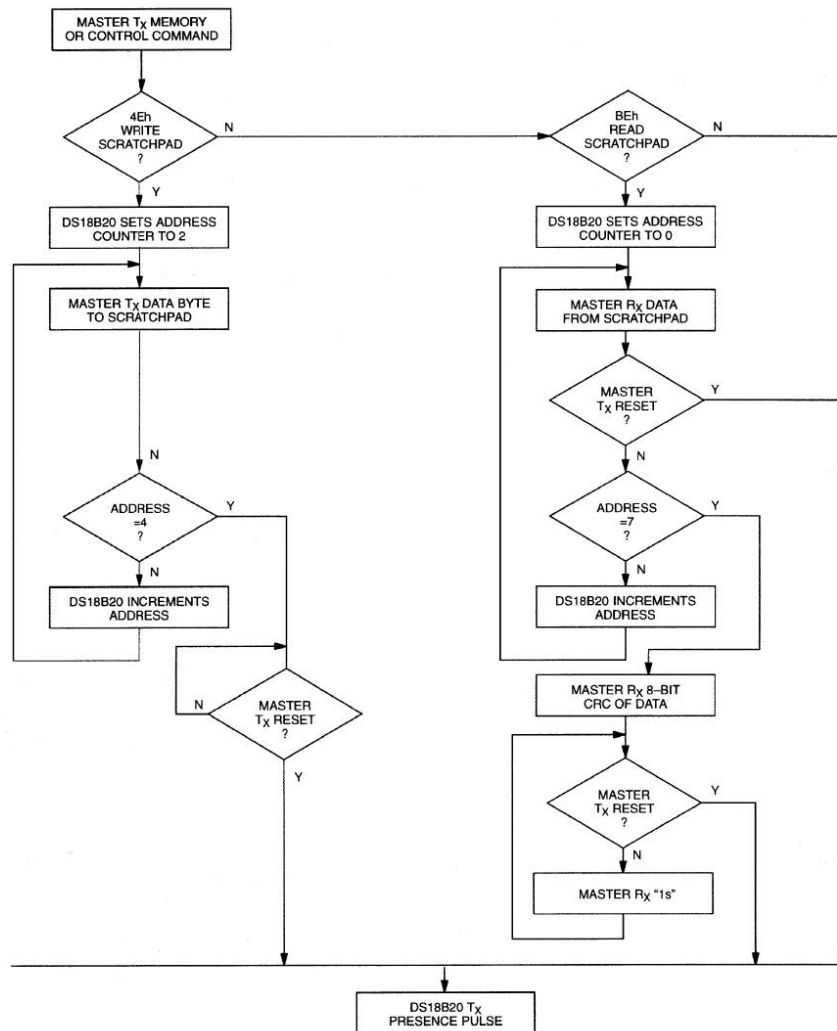
This command begins a temperature conversion. No further data is required. The temperature conversion will be performed and then the DS18B20 will remain idle. If the bus master issues read time slots following this command, the DS18B20 will output 0 on the bus as long as it is busy making a temperature conversion; it will return a 1 when the temperature conversion is complete. If parasite-powered, the bus master has to enable a strong pullup for a period greater than  $t_{conv}$  immediately after issuing this command.

### Recall E2 [B8h]

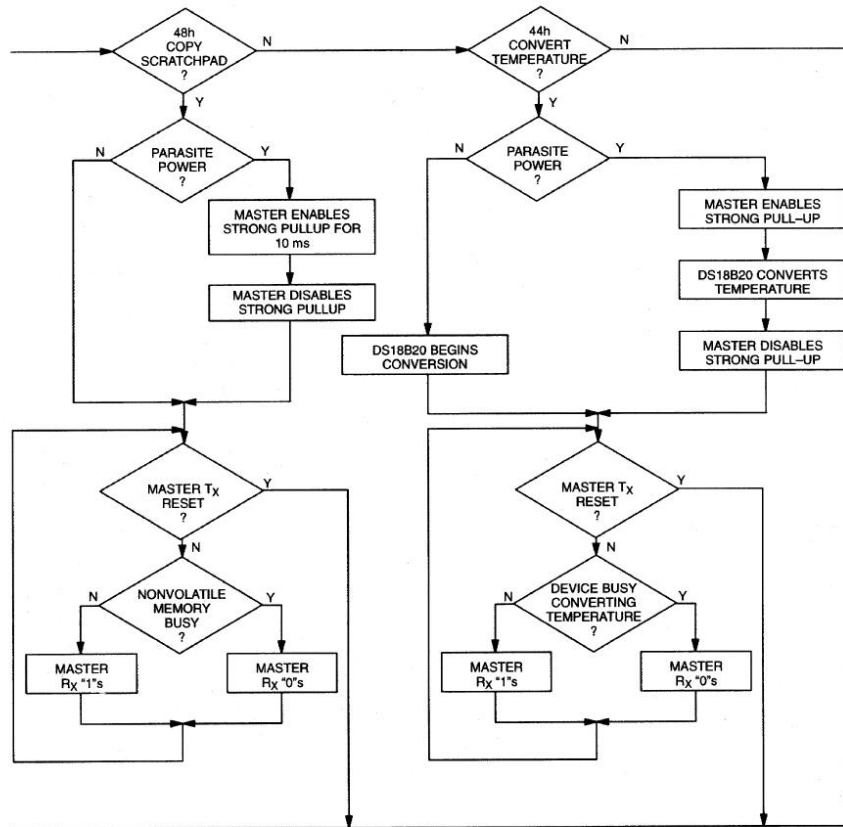
This command recalls the temperature trigger values and configuration register stored in E<sup>2</sup> to the scratchpad. This recall operation happens automatically upon power-up to the DS18B20 as well, so valid data is available in the scratchpad as soon as the device has power applied. With every read data time slot issued after this command has been sent, the device will output its temperature converter busy flag: 0=busy, 1=ready.

### Read Power Supply [B4h]

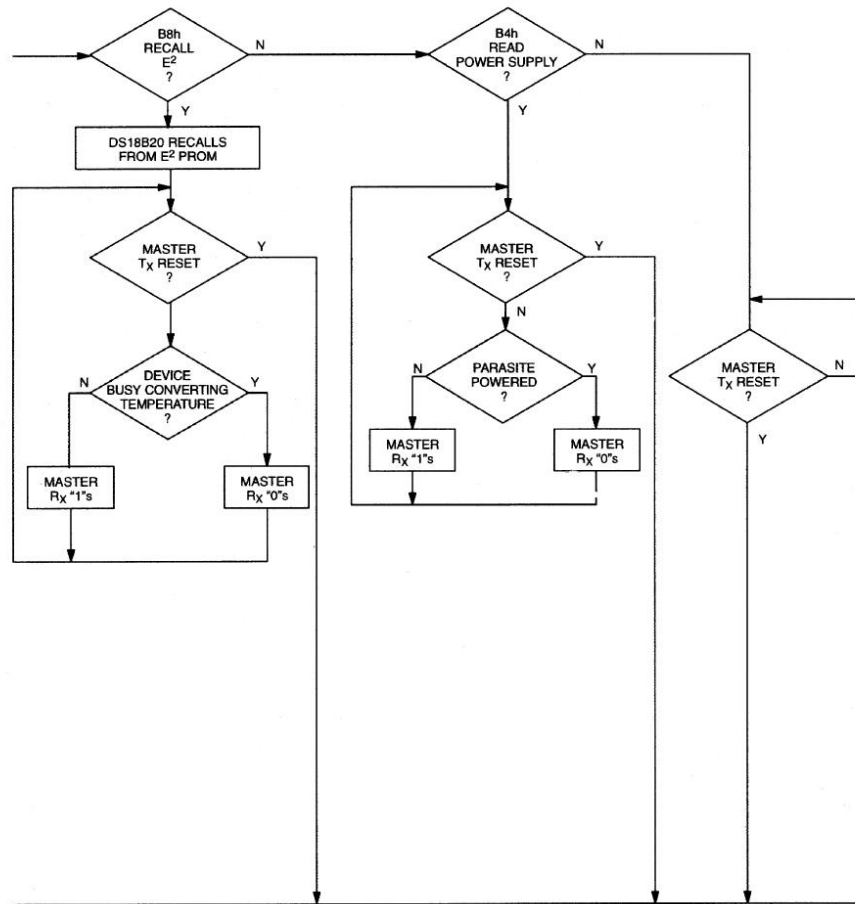
With every read data time slot issued after this command has been sent to the DS18B20, the device will signal its power mode: 0=parasite power, 1=external power supply provided.

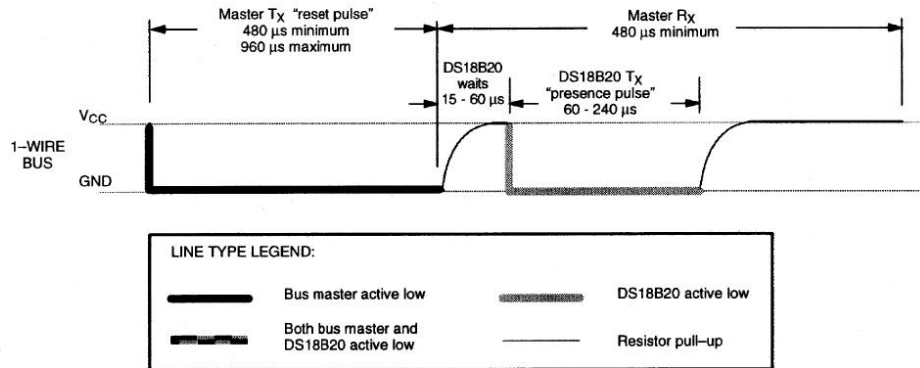
**MEMORY FUNCTIONS FLOW CHART** Figure 10

# **MEMORY FUNCTIONS FLOW CHART Figure 10 (cont'd)**



# **MEMORY FUNCTIONS FLOW CHART Figure 10 (cont'd)**



**INITIALIZATION PROCEDURE “RESET AND PRESENCE PULSES”** Figure 11**DS18B20 COMMAND SET** Table 4

| INSTRUCTION                            | DESCRIPTION                                                                                        | PROTOCOL | 1-WIRE BUS AFTER ISSUING PROTOCOL               | NOTES |
|----------------------------------------|----------------------------------------------------------------------------------------------------|----------|-------------------------------------------------|-------|
| <b>TEMPERATURE CONVERSION COMMANDS</b> |                                                                                                    |          |                                                 |       |
| Convert T                              | Initiates temperature conversion.                                                                  | 44h      | <read temperature busy status>                  | 1     |
| <b>MEMORY COMMANDS</b>                 |                                                                                                    |          |                                                 |       |
| Read Scratchpad                        | Reads bytes from scratchpad and reads CRC byte.                                                    | BEh      | <read data up to 9 bytes>                       |       |
| Write Scratchpad                       | Writes bytes into scratchpad at addresses 2 through 4 (TH and TL temperature triggers and config). | 4Eh      | <write data into 3 bytes at addr. 2 through. 4> | 3     |
| Copy Scratchpad                        | Copies scratchpad into nonvolatile memory (addresses 2 through 4 only).                            | 48h      | <read copy status>                              | 2     |
| Recall E <sup>2</sup>                  | Recalls values stored in nonvolatile memory into scratchpad (temperature triggers).                | B8h      | <read temperature busy status>                  |       |
| Read Power Supply                      | Signals the mode of DS18B20 power supply to the master.                                            | B4h      | <read supply status>                            |       |

**NOTES:**

1. Temperature conversion takes up to 750 ms. After receiving the Convert T protocol, if the part does not receive power from the  $V_{DD}$  pin, the DQ line for the DS18B20 must be held high for at least a period greater than  $t_{conv}$  to provide power during the conversion process. As such, no other activity may take place on the 1-Wire bus for at least this period after a Convert T command has been issued.
2. After receiving the Copy Scratchpad protocol, if the part does not receive power from the  $V_{DD}$  pin, the DQ line for the DS18B20 must be held high for at least 10 ms to provide power during the copy process. As such, no other activity may take place on the 1-Wire bus for at least this period after a Copy Scratchpad command has been issued.
3. All 3 bytes must be written before a reset is issued.

**READ/WRITE TIME SLOTS**

DS18B20 data is read and written through the use of time slots to manipulate bits and a command word to specify the transaction.

**Write Time Slots**

A write time slot is initiated when the host pulls the data line from a high logic level to a low logic level. There are two types of write time slots: Write 1 time slots and Write 0 time slots. All write time slots must be a minimum of 60  $\mu$ s in duration with a minimum of a 1- $\mu$ s recovery time between individual write cycles.

The DS18B20 samples the DQ line in a window of 15  $\mu$ s to 60  $\mu$ s after the DQ line falls. If the line is high, a Write 1 occurs. If the line is low, a Write 0 occurs (see Figure 12).

For the host to generate a Write 1 time slot, the data line must be pulled to a logic low level and then released, allowing the data line to pull up to a high level within 15  $\mu$ s after the start of the write time slot.

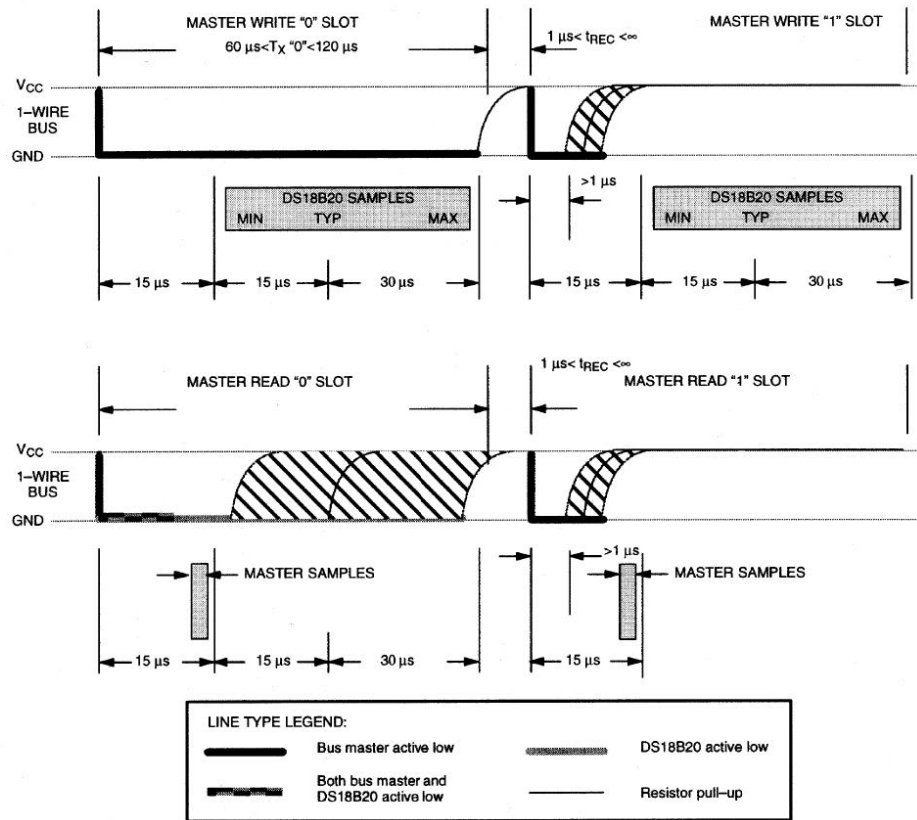
For the host to generate a Write 0 time slot, the data line must be pulled to a logic low level and remain low for 60  $\mu$ s.

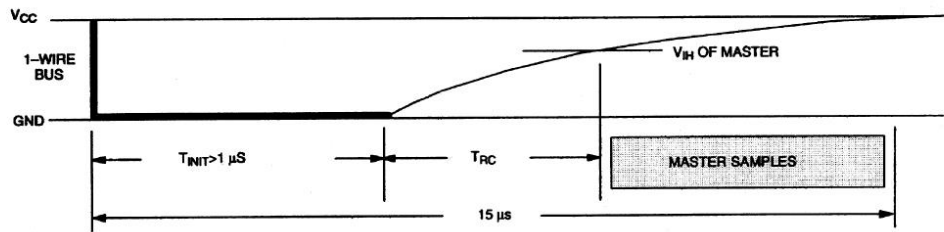
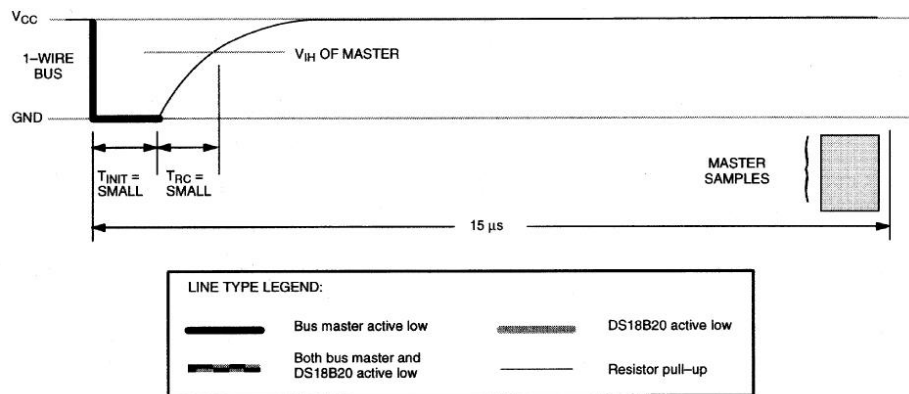
**Read Time Slots**

The host generates read time slots when data is to be read from the DS18B20. A read time slot is initiated when the host pulls the data line from a logic high level to logic low level. The data line must remain at a low logic level for a minimum of 1  $\mu$ s; output data from the DS18B20 is valid for 15  $\mu$ s after the falling edge of the read time slot. The host therefore must stop driving the DQ pin low in order to read its state 15  $\mu$ s from the start of the read slot (see Figure 12). By the end of the read time slot, the DQ pin will pull back high via the external pullup resistor. All read time slots must be a minimum of 60  $\mu$ s in duration with a minimum of a 1- $\mu$ s recovery time between individual read slots.

Figure 12 shows that the sum of  $T_{INIT}$ ,  $T_{RC}$ , and  $T_{SAMPLE}$  must be less than 15  $\mu$ s. Figure 14 shows that system timing margin is maximized by keeping  $T_{INIT}$  and  $T_{RC}$  as small as possible and by locating the master sample time towards the end of the 15- $\mu$ s period.



**READ/WRITE TIMING DIAGRAM** Figure 12

**DETAILED MASTER READ 1 TIMING Figure 13****RECOMMENDED MASTER READ 1 TIMING Figure 14**

### Related Application Notes

The following Application Notes can be applied to the DS18B20. These notes can be obtained from the Dallas Semiconductor "Application Note Book," via our website at <http://www.dalsemi.com/>.

Application Note 27: "Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Product"

Application Note 55: "Extending the Contact Range of Touch Memories"

Application Note 74: "Reading and Writing Touch Memories via Serial Interfaces"

Application Note 104: "Minimalist Temperature Control Demo"

Application Note 106: "Complex MicroLANs"

Application Note 108: "MicroLAN - In the Long Run"

Sample 1-Wire subroutines that can be used in conjunction with AN74 can be downloaded from the website or our Anonymous FTP Site.

### MEMORY FUNCTION EXAMPLE Table 5

Example: Bus Master initiates temperature conversion, then reads temperature (parasite power assumed).

| MASTER MODE | DATA (LSB FIRST)  | COMMENTS                                                                                                                                                                                                                                               |
|-------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TX          | Reset             | Reset pulse (480-960 $\mu$ s).                                                                                                                                                                                                                         |
| RX          | Presence          | Presence pulse.                                                                                                                                                                                                                                        |
| TX          | 55h               | Issue "Match ROM" command.                                                                                                                                                                                                                             |
| TX          | <64-bit ROM code> | Issue address for DS18B20.                                                                                                                                                                                                                             |
| TX          | 44h               | Issue "Convert T" command.                                                                                                                                                                                                                             |
| TX          | <I/O LINE HIGH>   | I/O line is held high for at least a period of time greater than $t_{conv}$ by bus master to allow conversion to complete.                                                                                                                             |
| TX          | Reset             | Reset pulse.                                                                                                                                                                                                                                           |
| RX          | Presence          | Presence pulse.                                                                                                                                                                                                                                        |
| TX          | 55h               | Issue "Match ROM" command.                                                                                                                                                                                                                             |
| TX          | <64-bit ROM code> | Issue address for DS18B20.                                                                                                                                                                                                                             |
| TX          | BEh               | Issue "Read Scratchpad" command.                                                                                                                                                                                                                       |
| RX          | <9 data bytes>    | Read entire scratchpad plus CRC; the master now recalculates the CRC of the eight data bytes received from the scratchpad, compares the CRC calculated and the CRC read. If they match, the master continues; if not, this read operation is repeated. |
| TX          | Reset             | Reset pulse.                                                                                                                                                                                                                                           |
| RX          | Presence          | Presence pulse, done.                                                                                                                                                                                                                                  |

### Related Application Notes

The following Application Notes can be applied to the DS18B20. These notes can be obtained from the Dallas Semiconductor "Application Note Book," via our website at <http://www.dalsemi.com/>.

Application Note 27: "Understanding and Using Cyclic Redundancy Checks with Dallas Semiconductor Touch Memory Product"

Application Note 55: "Extending the Contact Range of Touch Memories"

Application Note 74: "Reading and Writing Touch Memories via Serial Interfaces"

Application Note 104: "Minimalist Temperature Control Demo"

Application Note 106: "Complex MicroLANs"

Application Note 108: "MicroLAN - In the Long Run"

Sample 1-Wire subroutines that can be used in conjunction with AN74 can be downloaded from the website or our Anonymous FTP Site.

### MEMORY FUNCTION EXAMPLE Table 5

Example: Bus Master initiates temperature conversion, then reads temperature (parasite power assumed).

| MASTER MODE | DATA (LSB FIRST)  | COMMENTS                                                                                                                                                                                                                                               |
|-------------|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TX          | Reset             | Reset pulse (480-960 $\mu$ s).                                                                                                                                                                                                                         |
| RX          | Presence          | Presence pulse.                                                                                                                                                                                                                                        |
| TX          | 55h               | Issue "Match ROM" command.                                                                                                                                                                                                                             |
| TX          | <64-bit ROM code> | Issue address for DS18B20.                                                                                                                                                                                                                             |
| TX          | 44h               | Issue "Convert T" command.                                                                                                                                                                                                                             |
| TX          | <I/O LINE HIGH>   | I/O line is held high for at least a period of time greater than $t_{conv}$ by bus master to allow conversion to complete.                                                                                                                             |
| TX          | Reset             | Reset pulse.                                                                                                                                                                                                                                           |
| RX          | Presence          | Presence pulse.                                                                                                                                                                                                                                        |
| TX          | 55h               | Issue "Match ROM" command.                                                                                                                                                                                                                             |
| TX          | <64-bit ROM code> | Issue address for DS18B20.                                                                                                                                                                                                                             |
| TX          | BEh               | Issue "Read Scratchpad" command.                                                                                                                                                                                                                       |
| RX          | <9 data bytes>    | Read entire scratchpad plus CRC; the master now recalculates the CRC of the eight data bytes received from the scratchpad, compares the CRC calculated and the CRC read. If they match, the master continues; if not, this read operation is repeated. |
| TX          | Reset             | Reset pulse.                                                                                                                                                                                                                                           |
| RX          | Presence          | Presence pulse, done.                                                                                                                                                                                                                                  |

**MEMORY FUNCTION EXAMPLE Table 6**

Example: Bus Master writes memory (parasite power and only one DS18B20 assumed).

| MASTER MODE | DATA (LSB FIRST) | COMMENTS                                                                                                                                                                                                                                                             |
|-------------|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TX          | Reset            | Reset pulse.                                                                                                                                                                                                                                                         |
| RX          | Presence         | Presence pulse.                                                                                                                                                                                                                                                      |
| TX          | CCh              | Skip ROM command.                                                                                                                                                                                                                                                    |
| TX          | 4Eh              | Write Scratchpad command.                                                                                                                                                                                                                                            |
| TX          | <3 data bytes>   | Writes three bytes to scratchpad (TH, TL, and config).                                                                                                                                                                                                               |
| TX          | Reset            | Reset pulse.                                                                                                                                                                                                                                                         |
| RX          | Presence         | Presence pulse.                                                                                                                                                                                                                                                      |
| TX          | CCh              | Skip ROM command.                                                                                                                                                                                                                                                    |
| TX          | BEh              | Read Scratchpad command.                                                                                                                                                                                                                                             |
| RX          | <9 data bytes>   | Read entire scratchpad plus CRC. The master now recalculates the CRC of the eight data bytes received from the scratchpad, compares the CRC and the two other bytes read back from the scratchpad. If data match, the master continues; if not, repeat the sequence. |
| TX          | Reset            | Reset pulse.                                                                                                                                                                                                                                                         |
| RX          | Presence         | Presence pulse.                                                                                                                                                                                                                                                      |
| TX          | CCh              | Skip ROM command.                                                                                                                                                                                                                                                    |
| TX          | 48h              | Copy Scratchpad command; after issuing this command, the master must wait 10 ms for copy operation to complete.                                                                                                                                                      |
| TX          | Reset            | Reset pulse.                                                                                                                                                                                                                                                         |
| RX          | Presence         | Presence pulse, done.                                                                                                                                                                                                                                                |

**ABSOLUTE MAXIMUM RATINGS\***

|                                       |                              |
|---------------------------------------|------------------------------|
| Voltage on Any Pin Relative to Ground | -0.5V to +6.0V               |
| Operating Temperature                 | -55°C to +125°C              |
| Storage Temperature                   | -55°C to +125°C              |
| Soldering Temperature                 | See J-STD-020A specification |

\* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

**RECOMMENDED DC OPERATING CONDITIONS**

| PARAMETER      | SYMBOL          | CONDITION   | MIN  | TYP | MAX                  | UNITS | NOTES |
|----------------|-----------------|-------------|------|-----|----------------------|-------|-------|
| Supply Voltage | V <sub>DD</sub> | Local Power | 3.0  |     | 5.5                  | V     | 1     |
| Data Pin       | DQ              |             | -0.3 |     | +5.5                 | V     | 1     |
| Logic 1        | V <sub>IH</sub> |             | 2.2  |     | V <sub>CC</sub> +0.3 | V     | 1,2   |
| Logic 0        | V <sub>IL</sub> |             | -0.3 |     | +0.8                 | V     | 1,3,7 |

**DC ELECTRICAL CHARACTERISTICS** (-55°C to +125°C; V<sub>DD</sub>=3.0V to 5.5V)

| PARAMETER             | SYMBOL           | CONDITION             | MIN  | TYP | MAX  | UNITS | NOTES |
|-----------------------|------------------|-----------------------|------|-----|------|-------|-------|
| Thermometer Error     | t <sub>ERR</sub> | -10°C to +85°C        |      |     | ±½   | °C    |       |
|                       |                  | -55°C to +125°C       |      |     | ±2   |       |       |
| Input Logic High      | V <sub>IH</sub>  | Local Power           | 2.2  |     | 5.5  | V     | 1,2   |
|                       |                  | Parasite Power        | 3.0  |     |      | V     | 1,2   |
| Input Logic Low       | V <sub>IL</sub>  |                       | -0.3 |     | +0.8 | V     | 1,3,7 |
| Sink Current          | I <sub>L</sub>   | V <sub>IO</sub> =0.4V | -4.0 |     |      | mA    | 1     |
| Standby Current       | I <sub>DDs</sub> |                       |      | 750 | 1000 | nA    | 6,8   |
| Active Current        | I <sub>DD</sub>  |                       |      | 1   | 1.5  | mA    | 4     |
| DQ-Input Load Current | I <sub>DQ</sub>  |                       |      | 5   |      | µA    | 5     |

**AC ELECTRICAL CHARACTERISTICS: NV MEMORY**(-55°C to +125°C; V<sub>DD</sub>=3.0V to 5.5V)

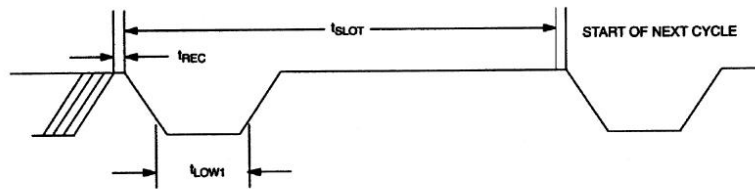
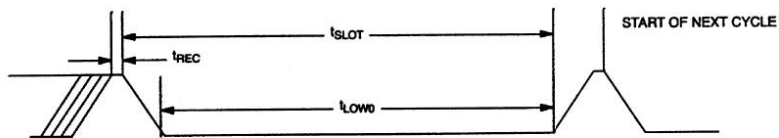
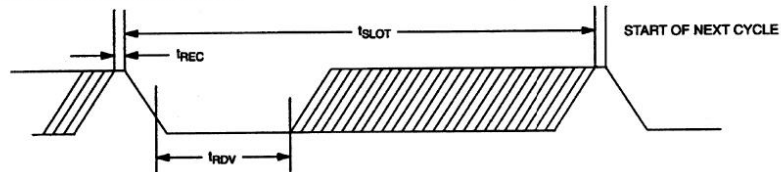
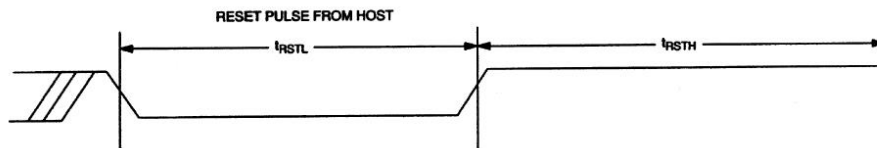
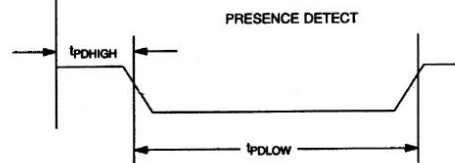
| PARAMETER             | SYMBOL            | CONDITION      | MIN | TYP | MAX | UNITS  | NOTES |
|-----------------------|-------------------|----------------|-----|-----|-----|--------|-------|
| NV Write Cycle Time   | t <sub>WT</sub>   |                |     | 2   | 10  | ms     |       |
| EEPROM Writes         | N <sub>EEWR</sub> | -55°C to +55°C | 50k |     |     | writes |       |
| EEPROM Data Retention | t <sub>EEDR</sub> | -55°C to +55°C | 10  |     |     | years  |       |

**AC ELECTRICAL CHARACTERISTICS:** (-55°C to +125°C;  $V_{DD}=3.0V$  to 5.5V)

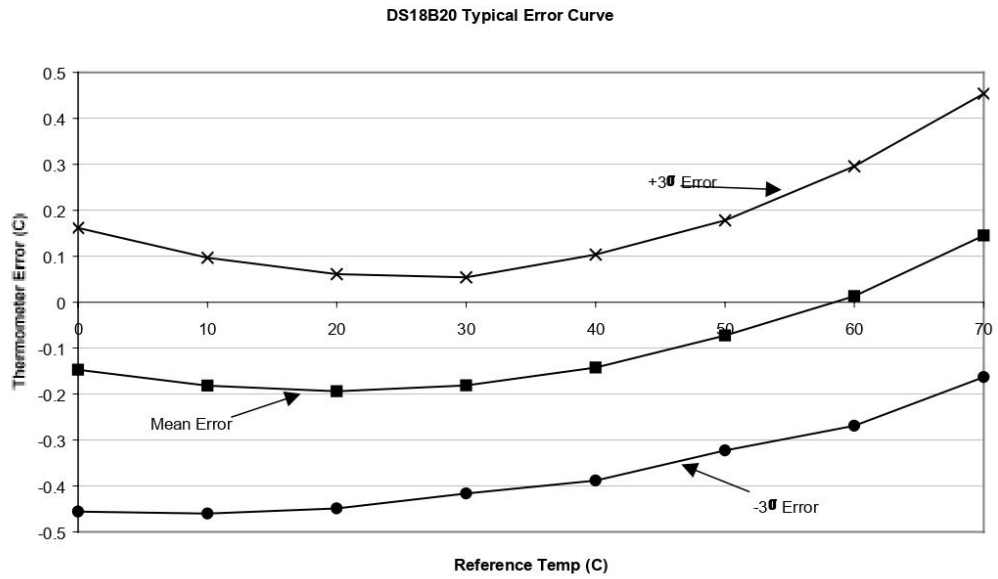
| PARAMETER                   | SYMBOL       | CONDITION | MIN | TYP | MAX   | UNITS   | NOTES |
|-----------------------------|--------------|-----------|-----|-----|-------|---------|-------|
| Temperature Conversion Time | $t_{CONV}$   | 9 bit     |     |     | 93.75 | ms      |       |
|                             |              | 10 bit    |     |     | 187.5 |         |       |
|                             |              | 11 bit    |     |     | 375   |         |       |
|                             |              | 12 bit    |     |     | 750   |         |       |
| Time Slot                   | $t_{SLOT}$   |           | 60  |     | 120   | $\mu s$ |       |
| Recovery Time               | $t_{REC}$    |           | 1   |     |       | $\mu s$ |       |
| Write 0 Low Time            | $t_{LOW0}$   |           | 60  |     | 120   | $\mu s$ |       |
| Write 1 Low Time            | $t_{LOW1}$   |           | 1   |     | 15    | $\mu s$ |       |
| Read Data Valid             | $t_{RDV}$    |           |     |     | 15    | $\mu s$ |       |
| Reset Time High             | $t_{RSTH}$   |           | 480 |     |       | $\mu s$ |       |
| Reset Time Low              | $t_{RSTL}$   |           | 480 |     |       | $\mu s$ | 9     |
| Presence Detect High        | $t_{PDHIGH}$ |           | 15  |     | 60    | $\mu s$ |       |
| Presence Detect Low         | $t_{PDLow}$  |           | 60  |     | 240   | $\mu s$ |       |
| Capacitance                 | $C_{IN/OUT}$ |           |     |     | 25    | pF      |       |

**NOTES:**

1. All voltages are referenced to ground.
2. Logic one voltages are specified at a source current of 1 mA.
3. Logic zero voltages are specified at a sink current of 4 mA.
4. Active current refers to either temperature conversion or writing to the E<sup>2</sup> memory. Writing to E<sup>2</sup> memory consumes approximately 200  $\mu A$  for up to 10 ms.
5. Input load is to ground.
6. Standby current specified up to 70°C. Standby current typically is 3  $\mu A$  at 125°C.
7. To always guarantee a presence pulse under low voltage parasite power conditions,  $V_{ILMAX}$  may have to be reduced to as much as 0.5V.
8. To minimize  $I_{DDs}$ , DQ should be:  $GND \leq DQ \leq GND + 0.3V$  or  $V_{DD} - 0.3V \leq DQ \leq V_{DD}$ .
9. Under parasite power, the max  $t_{RSTL}$  before a power on reset occurs is 960  $\mu s$ .

**1-WIRE WRITE ONE TIME SLOT****1-WIRE WRITE ZERO TIME SLOT****1-WIRE READ ZERO TIME SLOT****1-WIRE RESET PULSE****1-WIRE PRESENCE DETECT**



**TYPICAL PERFORMANCE CURVE**

## Anexo 2. Hoja de datos del sensor de flujo YF-S201.

Sensors & Sensor Modules : YF-S201 Water Flow Sensor

---

### YF-S201 Water Flow Sensor



This sensor sit in line with your water line, and uses a pinwheel sensor to measure how much liquid has moved through it.

Rating: Not Rated Yet

**Price:**

Variant price modifier:

Base price with tax:

Price with discount: 800,00 ?

Salesprice with discount:

Sales price: 800,00 ?

Sales price without tax: 800,00 ?

Discount:

Tax amount:

[Ask a question about this product](#)

Description

## Sensors & Sensor Modules : YF-S201 Water Flow Sensor

---

Measure liquid/water flow for your solar, water conservation systems, storage tanks, water recycling home applications, irrigation systems and much more. The sensors are solidly constructed and provide a digital pulse each time an amount of water passes through the pipe. The output can easily be connected to a microcontroller for monitoring water usage and calculating the amount of water remaining in a tank etc.

### Features:

- Model: YF-S201
- Working Voltage: 5 to 18V DC (min tested working voltage 4.5V)
- Max current draw: 15mA @ 5V
- Output Type: 5V TTL
- Working Flow Rate: 1 to 30 Liters/Minute
- Working Temperature range: -25 to +80°
- Working Humidity Range: 35%-80% RH
- Accuracy: ±10%
- Maximum water pressure: 2.0 MPa
- Output duty cycle: 50% ±10%
- Output rise time: 0.04us
- Output fall time: 0.18us
- Flow rate pulse characteristics:  $\text{Frequency (Hz)} = 7.5 * \text{Flow rate (L/min)}$
- Pulses per Liter: 450
- Durability: minimum 300,000 cycles
- Cable length: 15cm
- 1/2" nominal pipe connections, 0.78" outer diameter, 1/2" of thread
- Size: 2.5" x 1.4" x 1.4"

### ITEM INCLUDED:

1 x YF-S201 Water Flow Sensor

### Reviews

There are yet no reviews for this product.

## Anexo 3. Hoja de datos del motor a pasos NEMA 17.



17HS/42mm series 1.8 degree/step

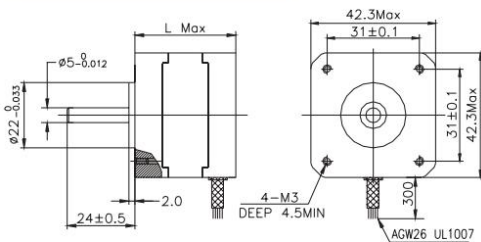


### Electrical Specification

| Model    | Step Angle (°) | Motor Length (mm) | Rated Current(A) | Phase Resistance(Ω) | Phase Inductance(mH) | Holding Torque (N.cmMin) | Detent Torque (N.cmMax) | Rotor Inertia (g.cm²) | Lead Wire (No.) | Motor Weight(g) |
|----------|----------------|-------------------|------------------|---------------------|----------------------|--------------------------|-------------------------|-----------------------|-----------------|-----------------|
| 17HS0410 | 1.8            | 20                | 0.8              | 3.8                 | 4.5                  | 10                       | 1.2                     | 24                    | 4               | 120             |
| 17HS2408 | 1.8            | 28                | 0.6              | 8                   | 10                   | 12                       | 1.6                     | 30                    | 4               | 150             |
| 17HS3401 | 1.8            | 34                | 1.3              | 2.4                 | 2.8                  | 28                       | 1.6                     | 34                    | 4               | 220             |
| 17HS3410 | 1.8            | 34                | 1.7              | 1.2                 | 1.8                  | 28                       | 1.6                     | 34                    | 4               | 220             |
| 17HS3630 | 1.8            | 34                | 0.4              | 30                  | 18                   | 21                       | 1.6                     | 34                    | 6               | 220             |
| 17HS4401 | 1.8            | 40                | 1.7              | 1.5                 | 2.8                  | 40                       | 2.2                     | 54                    | 4               | 280             |
| 17HS4402 | 1.8            | 40                | 1.3              | 2.5                 | 5.0                  | 40                       | 2.2                     | 54                    | 4               | 280             |
| 17HS4602 | 1.8            | 40                | 1.2              | 3.2                 | 2.8                  | 28                       | 2.2                     | 54                    | 6               | 280             |
| 17HS8401 | 1.8            | 48                | 1.7              | 1.8                 | 3.2                  | 52                       | 2.6                     | 68                    | 4               | 350             |
| 17HS8402 | 1.8            | 48                | 1.3              | 3.2                 | 5.5                  | 52                       | 2.6                     | 68                    | 4               | 350             |
| 17HS8403 | 1.8            | 48                | 2.3              | 1.2                 | 1.6                  | 46                       | 2.6                     | 68                    | 4               | 350             |
| 17HS8630 | 1.8            | 48                | 0.4              | 30                  | 38                   | 34                       | 2.6                     | 68                    | 6               | 350             |
| 17HS9403 | 1.8            | 60                | 2.3              | 1.4                 | 2.5                  | 70                       | 3.6                     | 80                    | 4               | 480             |
| 17HS6403 | 1.8            | 63                | 2.3              | 1.5                 | 2.2                  | 80                       | 4.0                     | 85                    | 4               | 500             |

\* Note: It's only typical technical data, special requirement can be customized.

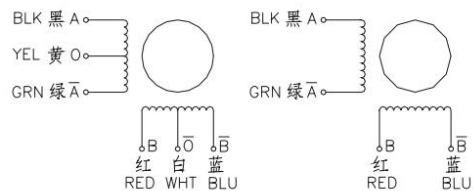
### Mechanical Dimensions



### Description:

| Model    | Motor Length |
|----------|--------------|
| 17HS2xxx | 28mm         |
| 17HS3xxx | 34mm         |
| 17HS4xxx | 40mm         |
| 17HS8xxx | 48mm         |
| 17HS9xxx | 58mm         |
| 17HS6xxx | 63mm         |

### Wiring Diagram

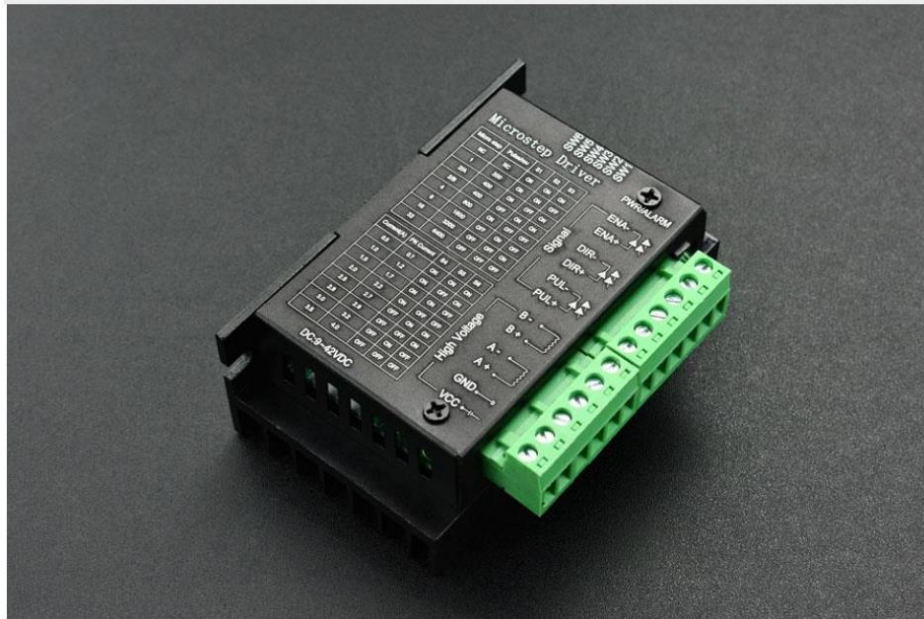


## Anexo 4. Hoja de datos del controlador TB6600.



### TB6600 Stepper Motor Driver

SKU:DRI0043



#### INTRODUCTION

##### What is a stepper motor?

[Stepper motor](#) is a brushless DC electric motor that divides a full rotation into a number of equal steps. The motor's position can then be commanded to move and hold at one of these steps.

##### How to control a stepper motor?

The fast way to control a stepper motor is just using a [stepper motor driver](#) (controller). And TB6600 arduino is just what you need.

TB6600 arduino is an easy-to-use professional stepper motor driver, which could control a two-phase stepping motor. It is compatible with [Arduino](#) and other microcontrollers that can output a 5V digital pulse signal. TB6600 arduino stepper motor driver has a wide range power input, 9~42VDC power supply. And it is able to output 4A peak current, which is enough for the most of [stepper motors](#).

The stepper driver supports speed and direction control. You can set its micro step and output current with 6 DIP switch. There are 7 kinds of micro steps (1, 2 / A, 2 / B, 4, 8, 16, 32) and 8 kinds of current control (0.5A, 1A, 1.5A, 2A, 2.5A, 2.8A, 3.0A, 3.5A) in all. And all signal terminals adopt high-speed optocoupler isolation, enhancing its anti-high-frequency interference ability.

As a professional device, it is able to drive 57, 42-type two-phase, four-phase, hybrid stepper motor.



## *FEATURES*

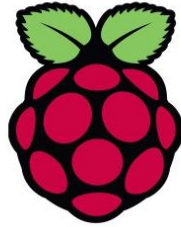
- Support 8 kinds of current control
- Support 7 kinds of micro steps adjustable
- The interface adopts high-speed optocoupler isolation
- Automatic semi-flow to reduce heat
- Large area heat sink
- Anti-high-frequency interference ability
- Input anti-reverse protection
- Overheat, over current and short circuit protection

## *SPECIFICATION*

- Input Current: 0~5A
- Output Current: 0.5~4.0A
- Control Signal: 3.3~24V
- Power (MAX): 160W
- Micro Step: 1, 2/A, 2/B, 4, 8, 16, 32
- Temperature: -10~45°C
- Humidity: No Condensation
- Dimension: 96\*56\*33 mm/ 3.78\*2.2\*1.3 inches
- Weight: 0.2 kg
- Drive IC: TB67S109AFTG

## **Anexo 5. Hoja de datos de la Raspberry Pi 4 Modelo B.**

# **DATASHEET**



## **Raspberry Pi 4 Model B**

**Release 1.1**

**March 2024**

**Copyright 2024 Raspberry Pi (Trading) Ltd. All rights reserved.**



Table 1: Release History

| Release | Date       | Description                                                 |
|---------|------------|-------------------------------------------------------------|
| 1       | 21/06/2019 | First release                                               |
| 1.1     | 12/03/2024 | Updated obsolescence statement and electrical specification |

The latest release of this document can be found at <https://www.raspberrypi.org>





## Contents

|                                        |           |
|----------------------------------------|-----------|
| <b>1 Introduction</b>                  | <b>5</b>  |
| <b>2 Features</b>                      | <b>6</b>  |
| 2.1 Hardware                           | 6         |
| 2.2 Interfaces                         | 6         |
| 2.3 Software                           | 6         |
| <b>3 Mechanical Specification</b>      | <b>7</b>  |
| <b>4 Electrical Specification</b>      | <b>7</b>  |
| 4.1 Power Requirements                 | 8         |
| <b>5 Peripherals</b>                   | <b>9</b>  |
| 5.1 GPIO Interface                     | 9         |
| 5.1.1 GPIO Pin Assignments             | 9         |
| 5.1.2 GPIO Alternate Functions         | 10        |
| 5.1.3 Display Parallel Interface (DPI) | 11        |
| 5.1.4 SD/SDIO Interface                | 11        |
| 5.2 Camera and Display Interfaces      | 11        |
| 5.3 USB                                | 11        |
| 5.4 HDMI                               | 11        |
| 5.5 Audio and Composite (TV Out)       | 11        |
| 5.6 Temperature Range and Thermals     | 11        |
| <b>6 Availability</b>                  | <b>11</b> |
| <b>7 Support</b>                       | <b>12</b> |



## List of Figures

|   |                            |   |
|---|----------------------------|---|
| 1 | Mechanical Dimensions      | 7 |
| 2 | Digital IO Characteristics | 8 |
| 3 | GPIO Connector Pinout      | 9 |



List of Tables

|   |                                         |    |
|---|-----------------------------------------|----|
| 1 | Release History                         | 1  |
| 2 | Absolute Maximum Ratings                | 7  |
| 3 | DC Characteristics                      | 8  |
| 4 | Digital I/O Pin AC Characteristics      | 8  |
| 5 | Raspberry Pi 4 GPIO Alternate Functions | 10 |



## 1 Introduction

The Raspberry Pi 4 Model B (Pi4B) is the first of a new generation of Raspberry Pi computers supporting more RAM and with significantly enhanced CPU, GPU and I/O performance in a similar form factor, power envelope and cost as the previous generation Raspberry Pi 3B+.

The Pi4B is available with 1, 2, 4, or 8 gigabytes of LPDDR4 SDRAM.



## 2 Features

### 2.1 Hardware

- Quad core 64-bit ARM-Cortex A72 running at 1.5GHz
- 1, 2 and 4 Gigabyte LPDDR4 RAM options
- H.265 (HEVC) hardware decode (up to 4Kp60)
- H.264 hardware decode (up to 1080p60)
- VideoCore VI 3D Graphics
- Supports dual HDMI display output up to 4Kp60

### 2.2 Interfaces

- 802.11 b/g/n/ac Wireless LAN
- Bluetooth 5.0 with BLE
- 1x SD Card
- 2x micro-HDMI ports supporting dual displays up to 4Kp60 resolution
- 2x USB2 ports
- 2x USB3 ports
- 1x Gigabit Ethernet port (supports PoE with add-on PoE HAT)
- 1x Raspberry Pi camera port (2-lane MIPI CSI)
- 1x Raspberry Pi display port (2-lane MIPI DSI)
- 28x user GPIO supporting various interface options:
  - Up to 6x UART
  - Up to 6x I2C
  - Up to 5x SPI
  - 1x SDIO interface
  - 1x DPI (Parallel RGB Display)
  - 1x PCM
  - Up to 2x PWM channels
  - Up to 3x GPCLK outputs

### 2.3 Software

- ARMv8 Instruction Set
- Mature Linux software stack
- Actively developed and maintained



- Recent Linux kernel support
- Many drivers upstreamed
- Stable and well supported userland
- Availability of GPU functions using standard APIs

### 3 Mechanical Specification

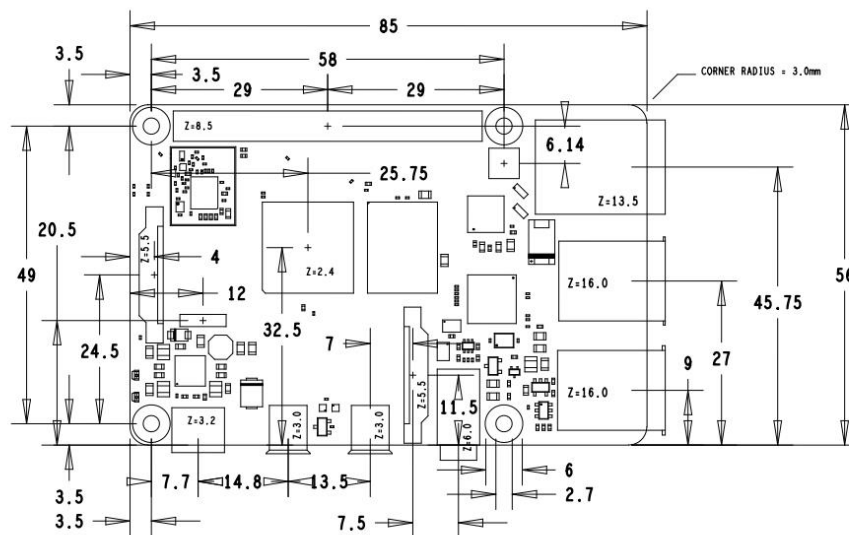


Figure 1: Mechanical Dimensions

### 4 Electrical Specification

**Caution!** Stresses above those listed in Table 2 may cause permanent damage to the device. This is a stress rating only; functional operation of the device under these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

| Symbol | Parameter        | Minimum | Maximum | Unit |
|--------|------------------|---------|---------|------|
| VIN    | 5V Input Voltage | -0.5    | 6.0     | V    |

Table 2: Absolute Maximum Ratings

Please note that VDD\_IO is the GPIO bank voltage which is tied to the on-board 3.3V supply rail.



| Symbol   | Parameter                        | Conditions                | Minimum      | Typical | Maximum | Unit |
|----------|----------------------------------|---------------------------|--------------|---------|---------|------|
| $V_{IL}$ | Input low voltage <sup>a</sup>   | VDD_IO = 3.3V             | 0            | -       | 0.8     | V    |
| $V_{IH}$ | Input high voltage <sup>a</sup>  | VDD_IO = 3.3V             | 2.0          | -       | VDD_IO  | V    |
| $I_{IL}$ | Input leakage current            | TA = +85°C                | -            | -       | 10      | μA   |
| $C_{IN}$ | Input capacitance                | -                         | -            | 5       | -       | pF   |
| $V_{OL}$ | Output low voltage <sup>b</sup>  | VDD_IO = 3.3V, IOL = -2mA | -            | -       | 0.4     | V    |
| $V_{OH}$ | Output high voltage <sup>b</sup> | VDD_IO = 3.3V, IOH = 2mA  | VDD_IO - 0.4 | -       | -       | V    |
| $I_{OL}$ | Output low current <sup>c</sup>  | VDD_IO = 3.3V, VO = 0.4V  | 7            | -       | -       | mA   |
| $I_{OH}$ | Output high current <sup>c</sup> | VDD_IO = 3.3V, VO = 2.3V  | 7            | -       | -       | mA   |
| $R_{PU}$ | Pullup resistor                  | -                         | 18           | 47      | 73      | kΩ   |
| $R_{PD}$ | Pulldown resistor                | -                         | 18           | 47      | 73      | kΩ   |

<sup>a</sup> Hysteresis enabled

<sup>b</sup> Default drive strength (8mA)

<sup>c</sup> Maximum drive strength (16mA)

Table 3: DC Characteristics

| Pin Name        | Symbol     | Parameter                     | Minimum | Typical | Maximum | Unit |
|-----------------|------------|-------------------------------|---------|---------|---------|------|
| Digital outputs | $t_{rise}$ | 10-90% rise time <sup>a</sup> | -       | TBD     | -       | ns   |
| Digital outputs | $t_{fall}$ | 90-10% fall time <sup>a</sup> | -       | TBD     | -       | ns   |

<sup>a</sup> Default drive strength, CL = 5pF, VDD\_IO = 3.3V

Table 4: Digital I/O Pin AC Characteristics



Figure 2: Digital IO Characteristics

#### 4.1 Power Requirements

The Pi4B requires a good quality USB-C power supply capable of delivering 5V at 3A. If attached downstream USB devices consume less than 500mA, a 5V, 2.5A supply may be used.



## 5 Peripherals

### 5.1 GPIO Interface

The Pi4B makes 28 BCM2711 GPIOs available via a standard Raspberry Pi 40-pin header. This header is backwards compatible with all previous Raspberry Pi boards with a 40-way header.

#### 5.1.1 GPIO Pin Assignments

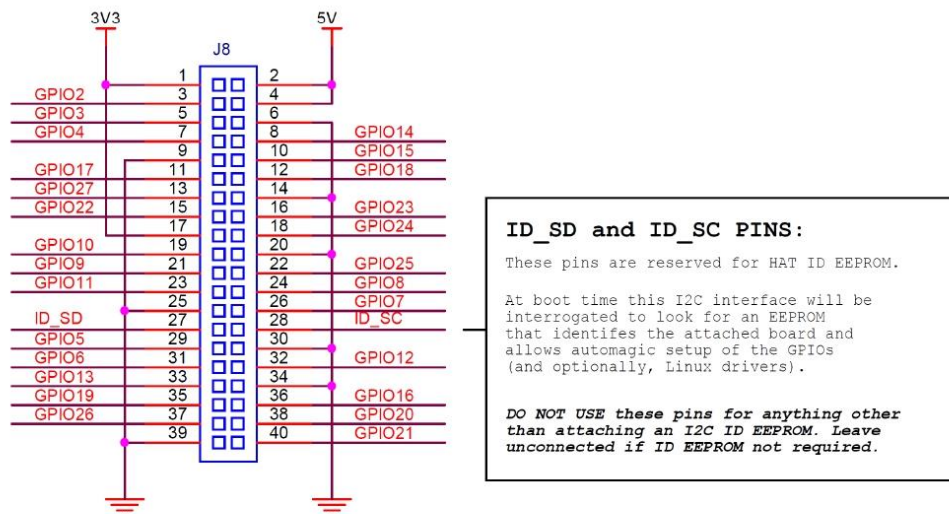


Figure 3: GPIO Connector Pinout

As well as being able to be used as straightforward software controlled input and output (with programmable pulls), GPIO pins can be switched (multiplexed) into various other modes backed by dedicated peripheral blocks such as I2C, UART and SPI.

In addition to the standard peripheral options found on legacy Pis, extra I2C, UART and SPI peripherals have been added to the BCM2711 chip and are available as further mux options on the Pi 4. This gives users much more flexibility when attaching add-on hardware as compared to older models.





### 5.1.2 GPIO Alternate Functions

| GPIO | Default Pull | ALT0       | ALT1  | ALT2      | ALT3       | ALT4       | ALT5       |
|------|--------------|------------|-------|-----------|------------|------------|------------|
| 0    | High         | SDA0       | SA5   | PCLK      | SPI3_CE0_N | TXD2       | SDA6       |
| 1    | High         | SCL0       | SA4   | DE        | SPI3_MISO  | RXD2       | SCL6       |
| 2    | High         | SDA1       | SA3   | LCD_VSYNC | SPI3_MOSI  | CTS2       | SDA3       |
| 3    | High         | SCL1       | SA2   | LCD_HSYNC | SPI3_SCLK  | RTS2       | SCL3       |
| 4    | High         | GPCLK0     | SA1   | DPLD0     | SPI4_CE0_N | TXD3       | SDA3       |
| 5    | High         | GPCLK1     | SA0   | DPLD1     | SPI4_MISO  | RXD3       | SCL3       |
| 6    | High         | GPCLK2     | SOE_N | DPLD2     | SPI4_MOSI  | CTS3       | SDA4       |
| 7    | High         | SPI0_CE1_N | SWE_N | DPLD3     | SPI4_SCLK  | RTS3       | SCL4       |
| 8    | High         | SPI0_CE0_N | SD0   | DPLD4     | -          | TXD4       | SDA4       |
| 9    | Low          | SPI0_MISO  | SD1   | DPLD5     | -          | RXD4       | SCL4       |
| 10   | Low          | SPI0_MOSI  | SD2   | DPLD6     | -          | CTS4       | SDA5       |
| 11   | Low          | SPI0_SCLK  | SD3   | DPLD7     | -          | RTS4       | SCL5       |
| 12   | Low          | PWM0       | SD4   | DPLD8     | SPI5_CE0_N | TXD5       | SDA5       |
| 13   | Low          | PWM1       | SD5   | DPLD9     | SPI5_MISO  | RXD5       | SCL5       |
| 14   | Low          | TXD0       | SD6   | DPLD10    | SPI5_MOSI  | CTS5       | TXD1       |
| 15   | Low          | RXD0       | SD7   | DPLD11    | SPI5_SCLK  | RTS5       | RXD1       |
| 16   | Low          | FL0        | SD8   | DPLD12    | CTS0       | SPI1_CE2_N | CTS1       |
| 17   | Low          | FL1        | SD9   | DPLD13    | RTS0       | SPI1_CE1_N | RTS1       |
| 18   | Low          | PCM_CLK    | SD10  | DPLD14    | SPI6_CE0_N | SPI1_CE0_N | PWM0       |
| 19   | Low          | PCM_FS     | SD11  | DPLD15    | SPI6_MISO  | SPI1_MISO  | PWM1       |
| 20   | Low          | PCM_DIN    | SD12  | DPLD16    | SPI6_MOSI  | SPI1_MOSI  | GPCLK0     |
| 21   | Low          | PCM_DOUT   | SD13  | DPLD17    | SPI6_SCLK  | SPI1_SCLK  | GPCLK1     |
| 22   | Low          | SD0_CLK    | SD14  | DPLD18    | SD1_CLK    | ARM_TRST   | SDA6       |
| 23   | Low          | SD0_CMD    | SD15  | DPLD19    | SD1_CMD    | ARM_RTCK   | SCL6       |
| 24   | Low          | SD0_DAT0   | SD16  | DPLD20    | SD1_DAT0   | ARM_TDO    | SPI3_CE1_N |
| 25   | Low          | SD0_DAT1   | SD17  | DPLD21    | SD1_DAT1   | ARM_TCK    | SPI4_CE1_N |
| 26   | Low          | SD0_DAT2   | TE0   | DPLD22    | SD1_DAT2   | ARM_TDI    | SPI5_CE1_N |
| 27   | Low          | SD0_DAT3   | TE1   | DPLD23    | SD1_DAT3   | ARM_TMS    | SPI6_CE1_N |

Table 5: Raspberry Pi 4 GPIO Alternate Functions

Table 5 details the default pin pull state and available alternate GPIO functions. Most of these alternate peripheral functions are described in detail in the BCM2711 Peripherals Specification document which can be downloaded from the [hardware documentation](#) section of the website.



### 5.1.3 Display Parallel Interface (DPI)

A standard parallel RGB (DPI) interface is available the GPIOs. This up-to-24-bit parallel interface can support a secondary display.

### 5.1.4 SD/SDIO Interface

The Pi4B has a dedicated SD card socket which supports 1.8V, DDR50 mode (at a peak bandwidth of 50 Megabytes / sec). In addition, a legacy SDIO interface is available on the GPIO pins.

## 5.2 Camera and Display Interfaces

The Pi4B has 1x Raspberry Pi 2-lane MIPI CSI Camera and 1x Raspberry Pi 2-lane MIPI DSI Display connector. These connectors are backwards compatible with legacy Raspberry Pi boards, and support all of the available Raspberry Pi camera and display peripherals.

## 5.3 USB

The Pi4B has 2x USB2 and 2x USB3 type-A sockets. Downstream USB current is limited to approximately 1.1A in aggregate over the four sockets.

## 5.4 HDMI

The Pi4B has 2x micro-HDMI ports, both of which support CEC and HDMI 2.0 with resolutions up to 4Kp60.

## 5.5 Audio and Composite (TV Out)

The Pi4B supports near-CD-quality analogue audio output and composite TV-output via a 4-ring TRS 'A/V' jack.

The analog audio output can drive 32 Ohm headphones directly.

## 5.6 Temperature Range and Thermals

The recommended ambient operating temperature range is 0 to 50 degrees Celsius.

To reduce thermal output when idling or under light load, the Pi4B reduces the CPU clock speed and voltage. During heavier load the speed and voltage (and hence thermal output) are increased. The internal governor will throttle back both the CPU speed and voltage to make sure the CPU temperature never exceeds 85 degrees C.

The Pi4B will operate perfectly well without any extra cooling and is designed for sprint performance - expecting a light use case on average and ramping up the CPU speed when needed (e.g. when loading a webpage). If a user wishes to load the system continually or operate it at a high temperature at full performance, further cooling may be needed.

## 6 Availability

Raspberry Pi guarantees availability of the Pi4B until at least January 2031.



## 7 Support

For support please see the hardware documentation section of the [Raspberry Pi website](#) and post questions to the [Raspberry Pi forum](#).

## **Anexo 6. Código de programación en Python del programa para la Raspberry pi.**

```
import os

import time

import gpiozero

import RPi.GPIO as GPIO

import threading

import matplotlib

import re

import tkinter as tk

import datetime

import numpy as np

import skfuzzy as fuzz

from flask import Flask, render_template, jsonify, request


from time import sleep

from threading import Thread

from skfuzzy import control as ctrl


app = Flask(__name__)


#algunas variables de la app

iniciar_sistema = False

#tempdes = ""

#tiempomin = ""

minut = 0

seg = 0
```

fecha = ""

hora = ""

#variables caudal

NumPulsos1 = 0

NumPulsos2 = 0

factor\_conversion = 7.5

frecuencia1 = 0

frecuencia2 = 0

#valor\_deseado=0.0

#valor\_deseado1=0.0

#valor\_deseado2=0.0

#valor\_deseado3=0.0

global temperatura\_sensada

global temperatura\_caliente

global temperatura\_fria

temperatura\_sensada=0.0

temperatura\_caliente=0.0

temperatura\_fria=0.0

#pines motores

step\_pincal = 17

dir\_pincal = 27

step\_pinfria = 23

dir\_pinfria = 24

```

#pines caudal

PinSensor1 = 5 # Primer caudalímetro
PinSensor2 = 6 # Segundo caudalímetro


#Gpio general
GPIO.setmode(GPIO.BCM)
GPIO.setup(step_pincal, GPIO.OUT)
GPIO.setup(dir_pincal, GPIO.OUT)
GPIO.setup(step_pinfria, GPIO.OUT)
GPIO.setup(dir_pinfria, GPIO.OUT)


#clase de caudalimetro
class Caudalimetro:
    def __init__(self):
        self.NumPulsos1 = 0
        self.NumPulsos2 = 0
        self.PinSensor1 = PinSensor1
        self.PinSensor2 = PinSensor2
        self.factor_conversion = factor_conversion
        self.lock1 = threading.Lock()
        self.lock2 = threading.Lock()
        self.caudal_L_m1=0.0
        self.caudal_L_m2=0.0
        self.frecuencia1=frecuencia1
        self.frecuencia2=frecuencia2
    def contar_pulsos1(self, channel):
        with self.lock1:

```

```

        self.NumPulsos1 += 1

def contar_pulsos2(self, channel):
    with self.lock2:
        self.NumPulsos2 += 1

def obtener_frecuencia1(self):
    with self.lock1:
        self.NumPulsos1 = 0
        GPIO.setup(self.PinSensor1, GPIO.OUT)
        GPIO.output(self.PinSensor1, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.setup(self.PinSensor1, GPIO.OUT)
        GPIO.output(self.PinSensor1, GPIO.LOW)
        GPIO.setup(PinSensor1, GPIO.IN)
        time.sleep(0.1)
    with self.lock1:
        self.frecuencia1 = self.NumPulsos1
    return self.NumPulsos1

def obtener_frecuencia2(self):
    with self.lock2:
        self.NumPulsos2 = 0
        GPIO.setup(self.PinSensor2, GPIO.OUT)
        GPIO.output(self.PinSensor2, GPIO.HIGH)
        time.sleep(0.1)
        GPIO.setup(self.PinSensor2, GPIO.OUT)

```

```

GPIO.output(self.PinSensor2, GPIO.LOW)
GPIO.setup(PinSensor2, GPIO.IN)
time.sleep(0.1)
with self.lock2:
    self.frecuencia2 = self.NumPulsos2
return self.NumPulsos2

def monitorear_caudalimetro1(self):
    while True:
        self.frecuencia1 = self.obtener_frecuencia1() - 1
        if self.frecuencia1 < 0:
            self.frecuencia1 = self.frecuencia1 + 1
        self.caudal_L_m1 = self.frecuencia1 / self.factor_conversion
        caudal_L_h1 = self.caudal_L_m1 * 60
        #print(f"Caudalímetro 1 - Frecuencia: {self.frecuencia1} Hz\tCaudal:
{self.caudal_L_m1:.3f} L/m\t{caudal_L_h1:.3f} L/h")
        time.sleep(1)
def obtener_caudal1(self):
    return self.caudal_L_m1

def monitorear_caudalimetro2(self):
    while True:
        self.frecuencia2 = self.obtener_frecuencia2() - 1
        if self.frecuencia2 < 0:
            self.frecuencia2 = self.frecuencia2 + 1
        self.caudal_L_m2 = self.frecuencia2 / self.factor_conversion
        caudal_L_h2 = self.caudal_L_m2 * 60

```



```
    #print(f"Caudalímetro 2 - Frecuencia: {self.frecuencia2} Hz\tCaudal:
{self.caudal_L_m2:.3f} L/m\t{caudal_L_h2:.3f} L/h")
```

```
    time.sleep(1)
```

```
def obtener_caudal2(self):
```

```
    return self.caudal_L_m2
```

```
#clase de sensores de temperatura
```

```
class DS18B20:
```

```
    def __init__(self, device_folder, name=None):
```

```
        self.device_folder = device_folder
```

```
        self.name = name
```

```
        self.temperature = 0.0
```

```
    def _read_temperature(self):
```

```
        try:
```

```
            with open(f"{self.device_folder}/w1_slave", 'r') as f:
```

```
                lines = f.readlines()
```

```
                if lines[0].strip()[-3:] == 'YES':
```

```
                    equals_pos = lines[1].find('t=')
```

```
                    if equals_pos != -1:
```

```
                        temp_str = lines[1][equals_pos+2:]
```

```
                        temp_c = float(temp_str) / 1000.0
```

```
                        #print (temp_c)
```

```
                        return temp_c
```

```
        except Exception as e:
```

```
            #print(f"Error reading temperature: {str(e)}")
```

```
            return None
```

```

def update_temperature(self):
    temp = self._read_temperature()
    if temp is not None:
        self.temperature = temp

#clase de reloj
class Reloj:
    def __init__(self):
        self.fecha = datetime.date.today()
        ahora = datetime.datetime.now()
        self.hora = ahora.strftime("%H:%M:%S")

    def actualizar_hora(self):
        self.fecha = datetime.date.today()
        ahora = datetime.datetime.now()
        self.hora = ahora.strftime("%H:%M:%S")

    def get_fecha(self):
        return self.fecha

    def get_hora(self):
        return self.hora

#clase de app
class NumericalKeyboard:
    def __init__(self, master, instancia_reloj, instancia_caudal, instancia_temperaturas,sensors):

```

```
self.instancia_temperaturas=instancia_temperaturas
```

```
self.instancia_caudal = instancia_caudal
```

```
self.instancia_reloj = instancia_reloj
```

```
self.sensors = sensors
```

```
self.master = master
```

```
self.temp_sensada= tk.StringVar()
```

```
self.temperatura_caliente= tk.StringVar()
```

```
self.temperatura_fria= tk.StringVar()
```

```
self.caudal1_tk=tk.StringVar()
```

```
self.caudal2_tk=tk.StringVar()
```

```
self.entry_var = tk.StringVar()
```

```
self.tempdes = tk.StringVar()
```

```
self.tiempomin = tk.StringVar()
```

```
self.tiemposeg = tk.StringVar()
```

```
self.fechita = tk.StringVar()
```

```
self.horita = tk.StringVar()
```

```
self.entry_var.set("")
```

```
self.tempdes.set("TEMP DESEADA: 0 °C")
```

```
self.tiempomin.set("00:00 min")
```

```
self.master.title("Selector de Temperatura")
```

```
self.entry = tk.Entry(master, textvariable=self.entry_var, font=('Arial', 16), justify='center')
```

```
self.minuto = tk.Label(master, textvariable=self.tiempomin, font=('Arial', 11), justify='right')
```

```

self.letrero = tk.Label(master, text="SELECCIONAR °C y TIEMPO", font=('Arial', 14),
justify='center')

#label fecha y hora

self.fecha_label = tk.Label(master, textvariable=self.fechita, font=('Arial', 12, 'italic'),
justify='center')

self.hora_label = tk.Label(master, textvariable=self.horita, font=('Arial', 12, 'italic'),
justify='center')

#label de caudalímetros

self.caud1 = tk.Label(master, text="CAUDAL CALIENTE:", font=('Arial', 12), justify='left')
self.caud2 = tk.Label(master, text="      CAUDAL FRIA:", font=('Arial', 12), justify='left')
self.caud1val = tk.Label(master, textvariable=self.caudal1_tk, font=('Arial', 12, 'underline'),
justify='left')
self.caud2val = tk.Label(master, textvariable=self.caudal2_tk, font=('Arial', 12, 'underline'),
justify='left')

self.caud1tx = tk.Label(master, text="L/m", font=('Arial', 12), justify='left')
self.caud2tx = tk.Label(master, text="L/m", font=('Arial', 12), justify='left')

#temperaturas label

self.temp2 = tk.Label(master, text="      TEMP SENSADA:", font=('Arial', 12), justify='right')
self.temp1VAL = tk.Label(master, textvariable=self.tempdes, font=('Arial', 11),
justify='center')

self.temp2VAL = tk.Label(master, textvariable=self.temp_sensada, font=('Arial', 12,
'underline'), justify='center')

self.temp2tx = tk.Label(master, text="°C", font=('Arial', 12), justify='center')
self.temp3 = tk.Label(master, text="      TEMP CALIENTE:", font=('Arial', 12), justify='right')
self.temp4 = tk.Label(master, text="      TEMP FRIA:", font=('Arial', 12), justify='right')
self.temp3VAL = tk.Label(master, textvariable=self.temperatura_caliente, font=('Arial', 12,
'underline'), justify='center')

self.temp4VAL = tk.Label(master, textvariable=self.temperatura_fria, font=('Arial', 12,
'underline'), justify='center')

self.temp3tx = tk.Label(master, text="°C", font=('Arial', 12), justify='center')

```

```

self.temp4tx = tk.Label(master, text="°C", font=('Arial', 12), justify='center')

#modular label

self.modup = tk.Label(master, text="SUBIR °C", font=('Arial', 8), justify='center')
self.moddn = tk.Label(master, text="BAJAR °C", font=('Arial', 8), justify='center')

#posicion barra de entrada y °C

self.entry.grid(row=1, column=0, columnspan=3, padx=1, pady=5, sticky='ew')
self.letrero.grid(row=0, column=0, columnspan=3, padx=5, pady=5, sticky='ew')
self.minuto.grid(row=2, column=5, columnspan=2, padx=1, pady=5, sticky='ew')

#posicion caudales

self.caud1.grid(row=3, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
self.caud2.grid(row=4, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
self.caud1val.grid(row=3, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
self.caud2val.grid(row=4, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
self.caud1tx.grid(row=3, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
self.caud2tx.grid(row=4, column=6, columnspan=1, padx=5, pady=5, sticky='ew')

#posicion temperaturas

self.temp2.grid(row=5, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
self.temp1VAL.grid(row=2, column=3, columnspan=1, padx=5, pady=5, sticky='ew')
self.temp2VAL.grid(row=5, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
self.temp2tx.grid(row=5, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
self.temp3.grid(row=6, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
self.temp4.grid(row=7, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
self.temp3VAL.grid(row=6, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
self.temp4VAL.grid(row=7, column=5, columnspan=1, padx=5, pady=5, sticky='ew')
self.temp3tx.grid(row=6, column=6, columnspan=1, padx=5, pady=5, sticky='ew')
self.temp4tx.grid(row=7, column=6, columnspan=1, padx=5, pady=5, sticky='ew')

#posicion modular temp

```

```

self.modup.grid(row=6, column=0, columnspan=1, padx=0, pady=1, sticky='ew')
self.moddn.grid(row=7, column=0, columnspan=1, padx=0, pady=1, sticky='ew')
#grid hora y fecha
self.fecha_label.grid(row=0, column=3, columnspan=2, padx=5, pady=5, sticky='ew')
self.hora_label.grid(row=0, column=5, columnspan=2, padx=5, pady=5, sticky='ew')

buttons = [
    ('7', 2, 0), ('8', 2, 1), ('9', 2, 2),
    ('4', 3, 0), ('5', 3, 1), ('6', 3, 2),
    ('1', 4, 0), ('2', 4, 1), ('3', 4, 2),
    ('.', 5, 0), ('0', 5, 1), ('←', 5, 2),
    ('▲', 6, 1), ('↑', 6, 2),
    ('▼', 7, 1), ('↓', 7, 2)
]

for (text, row, col) in buttons:
    if text == '←':
        button = tk.Button(master, text=text, font=('Arial', 15), command=self.clear_entry)
    else:
        button = tk.Button(master, text=text, font=('Arial', 15), command=lambda t=text:
self.on_button_click(t))
        button.grid(row=row, column=col, columnspan=1, padx=1, pady=1, sticky='nsew')

buttons1 = [
    ('SET °C', 1, 3), ('SET TIME', 1, 5),
]

```

```

for (text, row, col) in buttons1:

    button = tk.Button(master, text=text, font=('Arial', 12, 'bold'), command=lambda t=text:
self.on_button_click(t))

    button.grid(row=row, column=col, rowspan=1, columnspan=2, padx=1, pady=1,
sticky='nsew')


buttons2 = [

    ('INICIAR', 8, 5), ('CANCELAR', 8, 3),

]


for (text, row, col) in buttons2:

    button = tk.Button(master, text=text, font=('Arial Black', 14), command=lambda t=text:
self.on_button_click(t))

    button.grid(row=row, column=col, columnspan=2, padx=1, pady=1, sticky='nsew')

self.obtener_temperatura_deseada()

self.obtener_temperatura_deseada1()

self.obtener_temperaturas_juntas()

self.update_clock()

self.actualizar_caudal()


#funciones de fecha, hora y temporizador. estan sincronizados para evitar error
def obtener_temperatura_deseada(self):

    #global tempdes

    #print(self.tempdes.get())

    self.valor_deseado = self.tempdes.get()

    self.valor_deseado1 = re.findall(r'\d+\.\d+|\d+', self.valor_deseado)

    self.valor_deseado2 = ''.join(self.valor_deseado1)

    self.valor_deseado3 = float(self.valor_deseado2)

```

```

        #print ("dese3 ",self.valor_deseado3)

        self.master.after(1000, self.obtener_temperatura_deseada)

    def obtener_temperatura_deseada1(self):

        self.master.after(1000, self.obtener_temperatura_deseada1)

        #print(self.obtener_temperatura_deseada1())

        return self.valor_deseado3

    def obtener_temperaturas_juntas(self):

        self.instancia_temperaturas.obtener_temperatura_sensada(self.sensors)

        temperatura_sensada1 =
self.instancia_temperaturas.obtener_temperatura_sensada(self.sensors)

        self.instancia_temperaturas.obtener_temperatura_caliente(self.sensors)

        temperatura_caliente1 =
self.instancia_temperaturas.obtener_temperatura_caliente(self.sensors)

        self.instancia_temperaturas.obtener_temperatura_fria(self.sensors)

        temperatura_fria1 = self.instancia_temperaturas.obtener_temperatura_fria(self.sensors)

        self.temp_sensada.set(str(temperatura_sensada1))

        self.temperatura_caliente.set(str(temperatura_caliente1))

        self.temperatura_fria.set(str(temperatura_fria1))

        self.master.after(1000, self.obtener_temperaturas_juntas)

    def actualizar_caudal(self):

        self.instancia_caudal.obtener_caudal1()

        caudal1 = round((self.instancia_caudal.obtener_caudal1()),3)

        self.instancia_caudal.obtener_caudal2()

        caudal2 = round((self.instancia_caudal.obtener_caudal2()),3)

        self.caudal1_tk.set(str(caudal1))

        self.caudal2_tk.set(str(caudal2))

        self.master.after(1000, self.actualizar_caudal)

    def update_clock(self):

```



```

global iniciar_sistema

self.instancia_reloj.actualizar_hora()

fecha_valor = self.instancia_reloj.get_fecha()

hora_valor = self.instancia_reloj.get_hora()

self.fechita.set("Fecha: " + str(fecha_valor))

self.horita.set("Hora: " + str(hora_valor))

tiempo_restante = self.tiempomin.get()

numero_tiempo_restante = re.findall(r'\d+\.\d+|\d+', tiempo_restante)

minutos_rest=int(numero_tiempo_restante[0])

segundos_rest=int(numero_tiempo_restante[1])

tiempo_rest_en_seg=(minutos_rest*60)+segundos_rest

if tiempo_rest_en_seg > 0:

    if iniciar_sistema==True:

        tiempo_rest_en_seg -=1

    else:

        iniciar_sistema=False

        #print ("Sistema:False")

tiempo_decimal=tiempo_rest_en_seg/60

minutero = int(tiempo_decimal)

segundero = int(round(((tiempo_decimal - minutero)*60),1))

if segundero < 10:

    self.tiempomin.set(str(minutero) + ":0" + str(segundero) + " min")

else:

    self.tiempomin.set(str(minutero)+":"+str(segundero)+ " min")

self.master.after(1000, self.update_clock)

def on_button_click(self, value):

    global iniciar_sistema, minut, seg

```

```

if value == 'SET °C':
    try:
        result = eval(self.entry_var.get())
        self.entry_var.set(str(result))
        self.tempdes.set("TEMP DESEADA: "+ str(result)+" °C")
        #print (result)
        self.clear_entry()
    except:
        self.entry_var.set("Error, ingresa numero")
elif value == 'SET TIME':
    try:
        result = eval(self.entry_var.get())
        minut = int(result)
        segun = round((result - minut), 3)
        seg = int(segun * 60)
        if seg < 10:
            self.tiempomin.set(str(minut) + ":0" + str(seg) + " min")
        else:
            self.tiempomin.set(str(minut) + ":" + str(seg) + " min")
        #print (minut, ":", seg, " min")
        self.clear_entry()
        if minut == 0 and seg == 0:
            iniciar_sistema = False
            print ("Sistema: ", iniciar_sistema)
    except:
        self.entry_var.set("Error, ingresa numero")
elif value == 'INICIAR':

```

```

    iniciar_sistema = True

    print ("Sistema: ", iniciar_sistema)

elif value == 'CANCELAR':

    minut = 0

    seg = 0

    result = 0

    self.tiempomin.set("0" + str(minut) + ":0" + str(seg) + " min")

    self.tempdes.set("TEMP DESEADA: " + str(result)+" °C")

    self.clear_entry()

    iniciar_sistema = False

    #print ("Sistema: ", iniciar_sistema)

elif value == '←':

    self.clear_entry()

elif value == '↓':

    self.mdon = self.tempdes.get()

    numdn = re.findall(r'\d+\.?\d+|\d+', self.mdon)

    numa = ''.join(numdn)

    self.mdon_val = float(numa)

    self.mdon_val = self.mdon_val - 0.1

    don_red = round(self.mdon_val, 3)

    if don_red <= 0:

        don_red = 0

    #print (don_red)

    self.tempdes.set("TEMP DESEADA: " + str(don_red)+" °C")

    value = ''

    self.clear_entry()

elif value == '↑':

```

```

self.mup = self.tempdes.get()
numup = re.findall(r'\d+\.\d+|\d+', self.mup)
numb = ''.join(numup)
self.mup_val = float(numb)
self.mup_val = self.mup_val + 0.1
up_red = round(self.mup_val, 3)
#print (up_red)
self.tempdes.set("TEMP DESEADA: "+ str(up_red)+" °C")
value = ' '
self.clear_entry()
elif value == '▼':
    self.mdon = self.tempdes.get()
    numdn = re.findall(r'\d+\.\d+|\d+', self.mdon)
    numa = ''.join(numdn)
    self.mdon_val = float(numa)
    self.mdon_val -= 1
    if self.mdon_val < 0:
        self.mdon_val = 0
    #print (self.mdon_val)
    don_red = round(self.mdon_val, 3)
    self.tempdes.set("TEMP DESEADA: "+ str(don_red)+" °C")
    value = ' '
    self.clear_entry()
elif value == '▲':
    self.mup = self.tempdes.get()
    numup = re.findall(r'\d+\.\d+|\d+', self.mup)
    numb = ''.join(numup)

```

```

        self.mup_val = float(numb)
        self.mup_val += 1
        #print (self.mup_val)
        up_red = round(self.mup_val, 3)
        self.tempdes.set("TEMP DESEADA: "+ str(up_red)+" °C")
        value = ' '
        self.clear_entry()
    else:
        current_text = self.entry_var.get()
        if value == '.':
            if '.' not in current_text:
                self.entry_var.set(current_text + value)
            else:
                self.entry_var.set(current_text + value)

def clear_entry(self):
    self.entry_var.set("")

class FuzzyLogicController:
    def __init__(self):
        # Variables lingüísticas
        error = ctrl.Antecedent(np.arange(-50, 50, 0.01), 'error')
        cali = ctrl.Consequent(np.arange(0, 50, 1), 'cali')
        fria = ctrl.Consequent(np.arange(0, 50, 1), 'fria')
        # Funciones de pertenencia
        self._definir_funciones_pertenencia(error, cali, fria)
        # Reglas

```

```

self._definir_reglas(error, cali, fria)

# Sistema de control

self.cal_ctrl = ctrl.ControlSystem([self.rule1, self.rule2, self.rule3, self.rule4, self.rule5])
self.cali_ctrl = ctrl.ControlSystemSimulation(self.cal_ctrl)
self.fri_ctrl = ctrl.ControlSystem([self.rule6, self.rule7, self.rule8, self.rule9, self.rule10])
self.fria_ctrl = ctrl.ControlSystemSimulation(self.fri_ctrl)

def _definir_funciones_pertenencia(self, error, cali, fria):
    # Funciones de pertenencia

    error['erng'] = fuzz.trapmf(error.universe, [-50, -50, -2, -1.5])
    error['ernp'] = fuzz.trapmf(error.universe, [-2, -1.5, -0.02, 0])
    error['erz'] = fuzz.trimf(error.universe, [-0.02, 0, 0.02])
    error['erpp'] = fuzz.trapmf(error.universe, [0, 0.02, 1.5, 2])
    error['erpg'] = fuzz.trapmf(error.universe, [1.5, 2, 50, 50 ])

    cali['cerr'] = fuzz.trimf(cali.universe, [0, 0, 12.5])
    cali['poco'] = fuzz.trimf(cali.universe, [0, 12.5, 25])
    cali['med'] = fuzz.trimf(cali.universe, [12.5, 25, 37.5])
    cali['casi'] = fuzz.trimf(cali.universe, [25, 37.5, 50])
    cali['abi'] = fuzz.trimf(cali.universe, [37.5, 50, 50])

    fria['cerr'] = fuzz.trimf(fria.universe, [0, 0, 12.5])
    fria['poco'] = fuzz.trimf(fria.universe, [0, 12.5, 25])
    fria['med'] = fuzz.trimf(fria.universe, [12.5, 25, 37.5])
    fria['casi'] = fuzz.trimf(fria.universe, [25, 37.5, 50])
    fria['abi'] = fuzz.trimf(fria.universe, [37.5, 50, 50])

```

```

def _definir_reglas(self, error, cali, fria):
    # Reglas
    self.rule1 = ctrl.Rule(error['erng'], cali['cerr'])
    self.rule2 = ctrl.Rule(error['ernp'], cali['med'])
    self.rule3 = ctrl.Rule(error['erz'], cali['med'])
    self.rule4 = ctrl.Rule(error['erpp'], cali['abi'])
    self.rule5 = ctrl.Rule(error['erpg'], cali['abi'])
    self.rule6 = ctrl.Rule(error['erng'], fria['abi'])
    self.rule7 = ctrl.Rule(error['ernp'], fria['abi'])
    self.rule8 = ctrl.Rule(error['erz'], fria['med'])
    self.rule9 = ctrl.Rule(error['erpp'], fria['med'])
    self.rule10 = ctrl.Rule(error['erpg'], fria['cerr'])

```

```

def calcular_salida(self, er):
    self.cali_ctrl.input['error'] = er
    self.fria_ctrl.input['error'] = er

    # Cálculo difuso
    self.cali_ctrl.compute()
    self.fria_ctrl.compute()

    # Devolver resultados
    resultado_cali = self.cali_ctrl.output['cali']
    resultado_fria = self.fria_ctrl.output['fria']
    return resultado_cali, resultado_fria

```

```

class StepperControl:

```

```

def __init__(self, step_pincal, dir_pincal, step_pinfria, dir_pinfria):
    self.step_pincal = step_pincal
    self.dir_pincal = dir_pincal
    self.step_pinfria = step_pinfria
    self.dir_pinfria = dir_pinfria

    GPIO.setmode(GPIO.BCM)
    GPIO.setup(self.step_pincal, GPIO.OUT)
    GPIO.setup(self.dir_pincal, GPIO.OUT)
    GPIO.setup(self.step_pinfria, GPIO.OUT)
    GPIO.setup(self.dir_pinfria, GPIO.OUT)

def move_stepper1(self, steps, delay, clockwise_cal=True):
    if clockwise_cal:
        GPIO.output(self.dir_pincal, GPIO.HIGH)
        #print(f"Motor caliente sentido horario.")
    else:
        GPIO.output(self.dir_pincal, GPIO.LOW)
        #print(f"Motor caliente sentido antihorario.")
    #print(f"Caliente movio {steps} pasos.")
    for _ in range(steps):
        GPIO.output(self.step_pincal, GPIO.HIGH)
        time.sleep(delay)
        GPIO.output(self.step_pincal, GPIO.LOW)
        time.sleep(delay)

def move_stepper2(self, steps, delay, clockwise_fria=True):

```



```

if clockwise_fria:
    GPIO.output(self.dir_pinfria, GPIO.HIGH)
    #print(f"Motor fria sentido horario.")
else:
    GPIO.output(self.dir_pinfria, GPIO.LOW)
    #print(f"Motor caliente sentido antihorario.")
#print(f"Fria movio {steps} pasos.")
for _ in range(steps):
    GPIO.output(self.step_pinfria, GPIO.HIGH)
    time.sleep(delay)
    GPIO.output(self.step_pinfria, GPIO.LOW)
    time.sleep(delay)

class tres_temperaturas:
    def __init__(self,sensors):
        pass
    def obtener_temperatura_sensada(self,sensors):
        while True:
            for sensor in sensors:
                sensor.update_temperature()
            temperatures = [sensor.temperature for sensor in sensors]
            self.temperatura_sensada=temperatures[0]
            return self.temperatura_sensada
    def obtener_temperatura_caliente(self,sensors):
        while True:
            for sensor in sensors:
                sensor.update_temperature()

```

```

        temperatures = [sensor.temperature for sensor in sensors]
        self.temperatura_caliente=temperatures[1]
        return self.temperatura_caliente
def obtener_temperatura_fria(self,sensors):
    while True:
        for sensor in sensors:
            sensor.update_temperature()
            temperatures = [sensor.temperature for sensor in sensors]
            self.temperatura_fria=temperatures[2]
        return self.temperatura_fria

instancia_reloj=Reloj()
instancia_caudal=Caudalimetro()
#funcion principal

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/get_datetime')
def get_datetime():
    now = datetime.now()
    return jsonify(date=now.strftime("%Y-%m-%d"), time=now.strftime("%H:%M:%S"))

@app.route('/update_temp', methods=['POST'])
def update_temp():
    temp = request.json['temp']

```

```

return jsonify(temp=temp)

def main():

    fuzzy_logic_controller = FuzzyLogicController()
    stepper_control = StepperControl(17, 27, 23, 24)

    # Definir las direcciones de los sensores DS18B20
    sensor_addresses = [
        "/sys/bus/w1/devices/28-900cb30a6461",
        "/sys/bus/w1/devices/28-011914060ff6",
        "/sys/bus/w1/devices/28-94c1b30a6461"
    ]

    # Crear instancias de sensores DS18B20
    sensors = [DS18B20(address, name=f"sensor_{i+1}") for i, address in
enumerate(sensor_addresses)]

    instancia_temperaturas=tres_temperaturas(sensors)

    # Crear e iniciar el hilo para la lectura de temperaturas
    #temperature_thread = threading.Thread(target=read_temperatures, args=(sensors,))
    #temperature_thread.start()

    temperature_thread1 =
threading.Thread(target=instancia_temperaturas.obtener_temperatura_sensada,
args=(sensors,))

    temperature_thread1.start()

    temperature_thread2 =
threading.Thread(target=instancia_temperaturas.obtener_temperatura_caliente,
args=(sensors,))

    temperature_thread2.start()

```

```

    temperature_thread3 =
threading.Thread(target=instancia_temperaturas.obtener_temperatura_fria, args=(sensors,))

    temperature_thread3.start()

    #creacion de hilos caudal

    hilo1 = threading.Thread(target=instancia_caudal.monitorear_caudalimetro1)
    hilo2 = threading.Thread(target=instancia_caudal.monitorear_caudalimetro2)

    hilo1.start()
    hilo2.start()


    root = tk.Tk()

    #try:
    prev_c = 0
    move_c = 0
    prev_f = 0
    move_f = 0
    er = 0

    #valor_sensado=0.0


    app = NumericalKeyboard(root, instancia_reloj, instancia_caudal, instancia_temperaturas,
sensors)

    #obtener_temperatura_sensada(temperatura_sensada)

    def update():
        nonlocal prev_c, move_c, prev_f, move_f, er

        sensada= instancia_temperaturas.obtener_temperatura_sensada(sensors,)

        valor_deseado4 = app.obtener_temperatura_deseada1()

        #print ("dese4 ",valor_deseado4)

        valor_sensado = sensada

        #print("sensa ",valor_sensado)

```

```

er = valor_deseado4 - valor_sensado

#er=0

resultado_cali, resultado_fria = fuzzy_logic_controller.calcular_salida(er)

#print(f"Resultado para cali: {resultado_cali}")
#print(f"Resultado para fria: {resultado_fria}")

steps_c = int(resultado_cali)
#print (steps_c)
steps_f = int(resultado_fria)
#print(steps_f)
move_c = abs(steps_c - prev_c)
move_f = abs(steps_f - prev_f)
if iniciar_sistema== True:
    if (steps_c >= 0 and steps_c <= 50 and steps_f >= 0 and steps_f <= 50):
        thread1 = threading.Thread(target=stepper_control.move_stepper1, args=(move_c,
0.03, steps_c >= prev_c))
        thread2 = threading.Thread(target=stepper_control.move_stepper2, args=(move_f,
0.03, steps_f >= prev_f))
        #print ("sirve")

        thread1.start()
        thread2.start()
    else:
        print(f"Tus {steps_c} y {steps_f} pasos superan el movimiento. Te encuentras en el paso
{prev_c} y {prev_f}.")
    else:

```

```

steps_c=0
steps_f=0
move_c = abs(steps_c - prev_c)
move_f = abs(steps_f - prev_f)
if (steps_c >= 0 and steps_c <= 50 and steps_f >= 0 and steps_f <= 50):

    thread1 = threading.Thread(target=stepper_control.move_stepper1, args=(move_c,
0.03, steps_c >= prev_c))

    thread2 = threading.Thread(target=stepper_control.move_stepper2, args=(move_f,
0.03, steps_f >= prev_f))
    #print ("sirve")

    thread1.start()
    thread2.start()
else:
    print(f"Tus {steps_c} y {steps_f} pasos superan el movimiento. Te encuentras en el paso
{prev_c} y {prev_f}.")

    #move_c=0
    #move_f=0
    #print ("apagado")
    prev_c = steps_c
    prev_f = steps_f
    root.after(3000,update)

root.after(3000,update)

```

```

GPIO.setmode(GPIO.BCM)
GPIO.setup(PinSensor1, GPIO.OUT)
GPIO.setup(PinSensor2, GPIO.OUT)
GPIO.setup(PinSensor1, GPIO.IN)
GPIO.setup(PinSensor2, GPIO.IN)
GPIO.add_event_detect(PinSensor1, GPIO.RISING, callback=instancia_caudal.contar_pulsos1)
GPIO.add_event_detect(PinSensor2, GPIO.RISING, callback=instancia_caudal.contar_pulsos2)
root.mainloop()

#escritura de pasos en los motores
#temperature_thread.join()
temperature_thread1.join()
temperature_thread2.join()
temperature_thread3.join()
hilo1.join()
hilo2.join()
thread1.join()
thread2.join()
update_thread.join()

#except UnicodeDecodeError:
    #print("Error de decodificacion. Verifica la codificacion del archivo.")

#except ValueError as ve:
    #print(f"Error: {ve}. Escriba un numero por favor.")

#finally:
    GPIO.cleanup()

if __name__ == "__main__":
    main()

```

## Anexo 7. Código en HTML para visualizar la aplicación web.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Selector de Temperatura</title>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    .container {
      width: 70%;
      margin: auto;
    }

    .row {
      display: flex;
      justify-content: space-between;
      margin-bottom: 10px;
    }

    .column {
      flex: 1;
      text-align: center;
    }

    .column input, .column button {
      width: 100%;
      padding: 10px;
      margin: 5px 0;
    }

    .keyboard {
      display: flex;
      flex-wrap: wrap;
      justify-content: center;
      margin-top: 10px;
    }

    .key {
      width: 30%;
```



```

padding: 8px;
margin: 2px;
background-color: #f0f0f0;
border: 1px solid #ccc;
cursor: pointer;
text-align: center;
font-size: 16px;
user-select: none;
}
</style>
<script>
function insertNumber(num) {
    var inputField = document.getElementById("temperature");
    inputField.value += num;
}

function clearInput() {
    var inputField = document.getElementById("temperature");
    inputField.value = "";
}
</script>
</head>
<body>
<div class="container">
    <h1>Selector de Temperatura </h1>
    <h2>Fecha: {{ fecha }} | Hora: {{ hora }} </h2>
    <form action="{{ url_for('set_temperature') }}" method="post">
        <div class="row">
            <div class="column">
                <label for="temperature">Temperatura Deseada:</label>
                <input type="text" id="temperature" name="temperature" value="{{
temperatura_sensada }}">
            </div>
            <div class="column">
                <label for="time">Tiempo (min:seg):</label>
                <input type="text" id="time" name="time" value="{{ minut }}:{{ seg }}">
            </div>
        </div>
        <div class="row">
            <div class="column">
                <button type="submit" formaction="{{ url_for('start') }}">Iniciar</button>
            </div>
            <div class="column">
                <button type="submit" formaction="{{ url_for('cancel') }}">Cancelar</button>
            </div>
        </div>
    </form>
</div>

```

```
        </div>
    </div>
</form>
<h2>Estado Actual</h2>
<p>Temperatura Sensada: {{ temperatura_sensada }} °C</p>
<p>Temperatura Caliente: {{ temperatura_caliente }} °C</p>
<p>Temperatura Fría: {{ temperatura_fria }} °C</p>
<p>Caudal 1: {{ caudal1 }} L/m</p>
<p>Caudal 2: {{ caudal2 }} L/m</p>
</div>
</body>
</html>
```