LINEARIZATION OF A TEMPERATURE TRANSDUCER USING THE NEURAL NETWORK TECHNIQUE BASED ON A PROGRAMMABLE DIGITAL SYSTEM

LINEALIZACIÓN DE UN TRANSDUCTOR DE TEMPERATURA UTILIZANDO LA TÉCNICA DE REDES NEURONALES BASADO EN UN SISTEMA DIGITAL PROGRAMABLE

Mg. Mabel Rocio Díaz Pineda, PhD Jeison Marin Alfonso
Phd. Daniel Alexander Velasco Capacho,PhD Carlos Lizardo Corzo

Unidades Tecnológicas de Santander
Calle de los Estudiantes No. 9-82 Real de Minas. Bucaramanga, Santander, Colombia
Tel.: 57-7-6917700 Ext. 1120-1121
E-mail:mdiaz@correo.uts.edu.co;
jmarin@correo.uts.edu.co:dvelazco@correo.uts.edu.co;ccorzo@correo.uts.edu.co

Abstract: the present research work, evidence, the improvement of the temperature measurement system in blackberry crops in greenhouse, under controlled environments. For this, an instrument is implemented that, at a very low cost, registers a performance that guarantees an exact and precise measurement. The developed device senses the temperature by means of an NTC thermistor and by means of a signal conditioning circuit based on operational amplifiers a voltage of 0 to 5Vdc is obtained. Once the thermistor signal is conditioned, a programmable digital system capable of linearizing and delivering the temperature measurement displayed on a screen is entered. The algorithm takes essential parameters from a previous development carried out in Matlab through the technique of artificial neural networks

Keywords: linearization, neuronal network, sensor, temperature, transducer.

Resumen: El presente trabajo de investigación, evidencia, el mejoramiento del sistema de medida de temperatura en los cultivos de mora en invernadero, bajo ambientes controlados. Para ello, se implementará un instrumento que, a muy bajo costo, logre un desempeño tal que garantice una medida exacta y precisa. El dispositivo desarrollado sensa la temperatura mediante un termistor tipo NTC y mediante un circuito de acondicionamiento de señal a base de amplificadores operacionales, obtiene una tensión de 0 a 5Vdc. Una vez se acondiciona la señal del termistor, esta ingresa a un sistema digital programable capaz de linealizar y entregar la medida de temperatura visualizada en un display. El algoritmo toma parámetros esenciales de un desarrollo previo realizado en Matlab bajo la técnica de redes neuronales artificiales.

Palabras clave: Linealización, redes neuronales, sensor, temperatura, transductor**.**

1. INTRODUCION

Uno de los propósitos marcados en el plan institucional de las UTS es aportar desde la academia a la región santandereana, en lo relacionado con el fortalecimiento y la calidad de productos agrícolas como el cacao, café y mora.

Es por esto, que desde los diferentes programas académicos de la institución se han venido desarrollando variedad de proyectos de carácter tecnológico con los que se pretende tecnificar el cultivo de estas especies contribuyendo así con el crecimiento económico de familias productoras de la región.

El presente trabajo centra sus esfuerzos en una de las variables que influyen en el crecimiento de los plantines de mora: "la temperatura"; se plantea como objetivo principal la linealización de un transductor de temperatura tipo termistor por medio de redes neuronales artificiales, donde se busca desarrollar un dispositivo capaz de medir la temperatura en un rango de 8ºC a 20ºC.

Para lograr lo propuesto, se utiliza un sensor de temperatura tipo termistor NTC, un circuito de acondicionamiento de señal, un sistema digital programable (Arduino), un display LCD, un termohigrómetro y una cámara térmica. Se quiere entonces que la señal provista por el transductor (termistor) sea tratada por el circuito de acondicionamiento quien entrega una señal representativa en tensión en el rango de 0 a 5Vdc, de esta manera se obtiene una conversión análogo – digital de la variable (temperatura) necesaria para el interés principal de este trabajo, lograr linealizar la respuesta no lineal del transductor. La señal mencionada entregada por el circuito de acondicionamiento ingresa al sistema digital programable (Arduino) quien la procesa de acuerdo a la lógica programada previamente y linealiza la curva para finalmente mostrar en pantalla el valor correspondiente de temperatura.

2. METODOLOGÍA Y DESARROLLO

El propósito de este proyecto es garantizar la confiabilidad de la temperatura obtenida por medio de un termistor el cual es un sensor de temperatura de bajo costo, y esto es dado comparándola contra el dispositivo patrón de temperatura.

El termistor genera un valor variable de resistencia en la salida por cada valor de temperatura que mida, lo primero que se hace es un circuito de acondicionamiento con el cual en su salida se obtiene un valor de voltaje que será la entrada para el sistema digital programable (Fig. 1).

Para este proyecto se utiliza el sistema digital programable Arduino, que provee el convertidor análogo/digital que viene especificado para los rangos de voltaje entre 0 y 5 voltios.

La resolución de la placa Arduino Mega contiene un convertidor analógico/digital multicanal de 10 bits. Esto significa que asignará los voltajes de entrada entre 0 y 5V en valores enteros entre 0 y 1023. Esto produce una resolución entre lecturas de: 5 voltios / 1024 unidades o 0.0049 voltios (4.9 mV) por unidad.

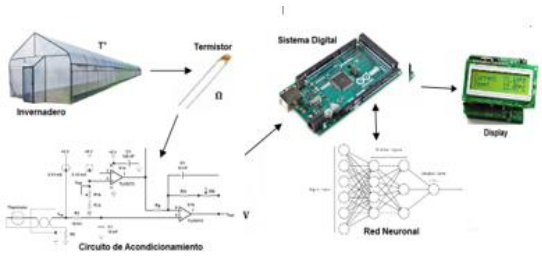

Fig. 1.Etapas para la obtener un termistor linealizado

Para este proyecto se utiliza el sistema digital programable Arduino, que provee el convertidor análogo/digital que viene especificado para los rangos de voltaje entre 0 y 5 voltios.

La resolución de la placa Arduino Mega contiene un convertidor analógico/digital multicanal de 10 bits. Esto significa que asignará los voltajes de entrada entre 0 y 5V en valores enteros entre 0 y 1023. Esto produce una resolución entre lecturas de: 5 voltios / 1024 unidades o 0.0049 voltios (4.9 mV) por unidad.

2.1 Circuito de acondicionamiento

La etapa del circuito de acondicionamiento para el termistor es fundamental dado que es acá donde se obtiene la señal de tensión representativa para cada valor de temperatura al que es expuesto el termistor, para ello se implementa el circuito de acondicionamiento mostrado en la Fig. 2.

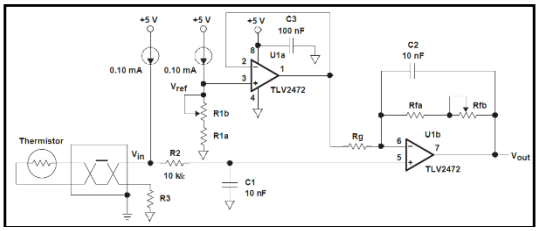

Fig. 2. Circuito de acondicionamiento

Este circuito es diseñado con base en amplificadores operacionales en configuración no-inversora con alta impedancia de entrada que garantizan por una parte la utilización de menos componentes y por otra, dada su alta impedancia, que cualquier corriente utilizada por la entrada del amplificador no afecte la señal medida. Este circuito utiliza dos fuentes de corriente que aseguran la misma intensidad para cualquier cambio de resistencia en el termistor y una tensión de referencia (Vref) para la entrada no inversora del amplificador operacional configurado como seguidor de tensión.

Las ecuaciones para conseguir los valores de resistencias y capacitores, vienen dadas por en su reporte de aplicación y las cuales se desarrollan para el uso de un termistor (Bishop , 2000) , con lo cual se obtienen los valores mostrados en la ecuación 1.

$$F_c = \frac{1}{2\pi R_2 C_1} \rightarrow R_2 = 10k\Omega \; ; \rightarrow R_G = 10nF \qquad (1)$$

Donde F_C es la frecuencia de corte del filtro RC. La ecuación (1) define la ganancia del circuito que viene dada por parámetros ya conocidos, como son el voltaje de salida mínimo y máximo, así como valores de resistencia del termistor en temperaturas mínimas y máximas empleadas en este proyecto. La ganancia del circuito es en esencia la relación entre el rango de voltaje de salida y el rango de voltaje de entrada.

$$m = \frac{Output_{máx} - Output_{mín}}{R_{0°C} * I_{Sensor} - R_{25°C} * I_{Sensor}}$$

$$m = \frac{5V - 0V}{32554\Omega * 0.1mA - 10000\Omega * 0.1mA} \qquad (1)$$

$$m = 2.217$$

Una vez calculada la ganancia del circuito y definiendo R_G en 26.7kΩ, la resistencia R_F de la Fig. 2 se puede obtener a partir de la ecuación (3), la cual define la ganancia particular para el amplificador operacional U1b.

$$|m| = \frac{R_F}{R_G} + 1 \quad \rightarrow \quad R_G = 26.7K\Omega$$

$$R_F = R_G(|m| - 1) = 26.7K\Omega(2.217 - 1) \cong 32.5K\Omega \qquad (2)$$

Para calcular V_{Ref} basta con sumar las caídas de voltaje en el termistor ($R_{25°C}I_{Sensor}$) y la resistencia R3, como se muestra en (4).

$$V_{Ref} = R_{25°C}I_{Sensor} + R_3 * I_{Sensor} \rightarrow \quad R_3 = 1K\Omega$$

$$V_{Ref} = 10K\Omega * 0.1mA + 1K\Omega * 0.1mA = 1.1V \qquad (4)$$

Con V_{Ref} calculado, aplicando ley de Ohm se obtiene R_1, como se muestra en (5).

$$R_1 = \frac{V_{Ref}}{I_{V_{Ref}}} = \frac{1.1V}{0.1mA} = 11K\Omega \qquad (5)$$

El circuito de acondicionamiento tiene una aplicación de larga vida, por lo que se pueden hacer los cálculos mostrados en las ecuaciones (6) a la (9).

$$R_{FA} = R_F * 0.91 = 32.4\,K\Omega * 0.91 \cong 29.4K\Omega \qquad (6)$$

$$R_{FB} = R_F * 2 * 0.09 = 32.4\,K\Omega * 2 * 0.09 \cong 5.86K\Omega \qquad (7)$$

$$R_{1A} = R_1 * 0.91 = 11\,K\Omega * 0.91 \cong 10K\Omega \qquad (8)$$

$$R_{1B} = R_1 * 2 * 0.09 = 11K\Omega * 2 * 0.09 \cong 2K\Omega \qquad (9)$$

Al obtener los valores de resistencias y capacitores se procede al montaje de la PCB donde funcionará el circuito de acondicionamiento para el termistor. En la Fig. 3 se muestra el diseño final de la PCB logrado teniendo en cuenta que se utilizan resistores y capacitores SMD, trimmers, conectores machos JST serie PH, y los circuitos integrados REF200 y TLV2472; listo para fabricación.

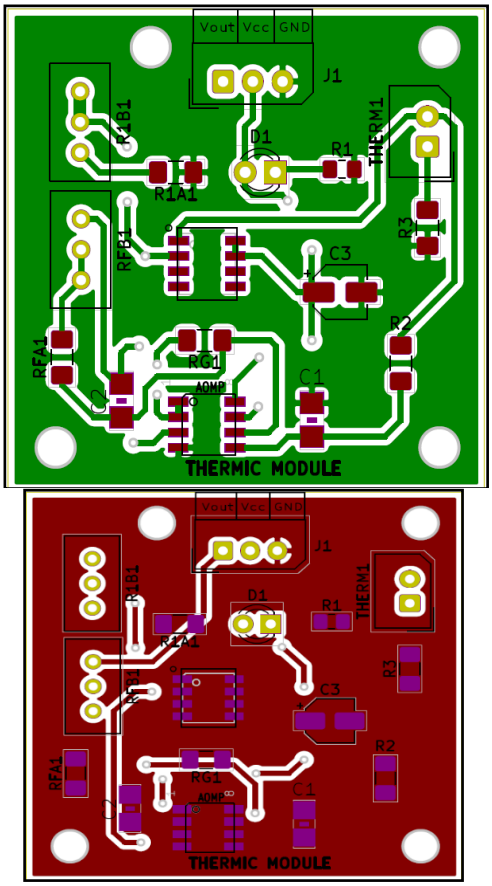

Fig. 3. Esquema KiCAD (top and back bottom)

En la Fig. 4 se muestra el montaje final de los elementos en la PCB.

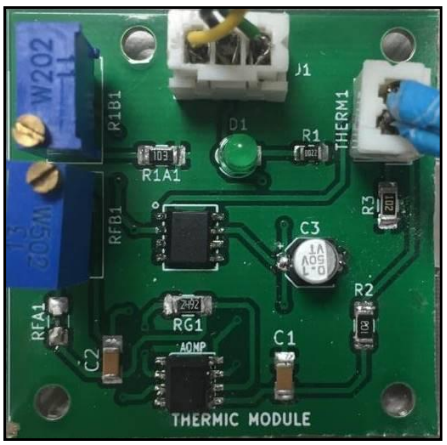

Fig. 4 Circuito de acondicionamiento ensamblado

2.2 Caracterización del sensor

El desarrollo de esta etapa del proyecto consiste en recolectar los datos necesarios que servirán como base para la creación de la red neuronal artificial y posterior linealización del modelo. Para esto, se varía la temperatura desde 25ºC a 5ºC (viceversa) una y otra vez en la cámara térmica, con ello se logra que el termistor (ubicado dentro de la cámara) actúe según la naturaleza propia de su funcionamiento variando su resistencia interna y con ello obtener los valores en tensión de la placa del circuito de acondicionamiento, esta tensión es ingresada al Arduino quien lo traduce a un valor de voltaje digital (Fig.5). Así mismo, en la Tabla 1 se obtiene la caracterización del sensor.

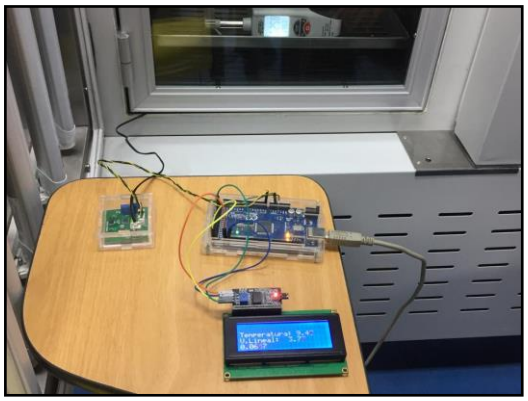

Fig. 5 Lectura de datos

Tabla 1: Datos obtenidos en la caracterización del sensor

T [ºC]	Volt	T [ºC]	Volt	T [ºC]	Volt	T [ºC]	Volt
4.5	4.97	**5.4**	4.76	**6.3**	4.51	**7.2**	4.28
4.6	4.95	**5.5**	4.73	**6.4**	4.47	**7.3**	4.23
4.7	4.94	**5.6**	4.71	**6.5**	4.45	**7.4**	4.19
4.8	4.93	**5.7**	4.70	**6.6**	4.42	**7.5**	4.16
4.9	4.91	**5.8**	4.65	**6.7**	4.40	**7.6**	4.15
5.0	4.90	**5.9**	4.63	**6.8**	4.37	**7.7**	4.13
5.1	4.87	**6.0**	4.60	**6.9**	4.35	**7.8**	4.11
5.2	4.82	**6.1**	4.58	**7.0**	4.33	**7.9**	4.07
5.3	4.77	**6.2**	4.54	**7.1**	4.31		
8.0	4.04	**11.1**	3.33	**14.2**	2.71	**17.3**	2.20
8.1	4.02	**11.2**	3.31	**14.3**	2.70	**17.4**	2.18
8.2	3.99	**11.3**	3.27	**14.4**	2.68	**17.5**	2.17
8.3	3.97	**11.4**	3.26	**14.5**	2.66	**17.6**	2.15
8.4	3.95	**11.5**	3.24	**14.6**	2.64	**17.7**	2.14
8.5	3.91	**11.6**	3.22	**14.7**	2.63	**17.8**	2.12
8.6	3.90	**11.7**	3.20	**14.8**	2.61	**17.9**	2.11
8.7	3.88	**11.8**	3.18	**14.9**	2.59	**18.0**	2.09
8.8	3.85	**11.9**	3.16	**15.0**	2.57	**18.1**	2.08
8.9	3.84	**12.0**	3.13	**15.1**	2.56	**18.2**	2.06
9.0	3.80	**12.1**	3.12	**15.2**	2.54	**18.3**	2.05
9.1	3.78	**12.2**	3.10	**15.3**	2.52	**18.4**	2.03
9.2	3.77	**12.3**	3.09	**15.4**	2.50	**18.5**	2.02
9.3	3.73	**12.4**	3.08	**15.5**	2.49	**18.6**	2.00
9.4	3.72	**12.5**	3.05	**15.6**	2.47	**18.7**	1.99
9.5	3.69	**12.6**	3.03	**15.7**	2.45	**18.8**	1.98
9.6	3.68	**12.7**	3.01	**15.8**	2.44	**18.9**	1.96
9.7	3.65	**12.8**	2.99	**15.9**	2.42	**19.0**	1.95
9.8	3.63	**12.9**	2.98	**16.0**	2.40	**19.1**	1.93
9.9	3.61	**13.0**	2.95	**16.1**	2.39	**19.2**	1.92
10.0	3.59	**13.1**	2.93	**16.2**	2.37	**19.3**	1.91
10.1	3.56	**13.2**	2.91	**16.3**	2.35	**19.4**	1.89
10.2	3.54	**13.3**	2.89	**16.4**	2.34	**19.5**	1.88
10.3	3.52	**13.4**	2.86	**16.5**	2.32	**19.6**	1.86
10.4	3.50	**13.5**	2.85	**16.6**	2.31	**19.7**	1.85
10.5	3.47	**13.6**	2.83	**16.7**	2.29	**19.8**	1.84
10.6	3.44	**13.7**	2.80	**16.8**	2.27	**19.9**	1.82
10.7	3.43	**13.8**	2.79	**16.9**	2.26	**20.0**	1.81
10.8	3.4	**13.9**	2.77	**17.0**	2.24		
10.9	3.37	**14.0**	2.75	**17.1**	2.23		
11.0	3.35	**14.1**	2.73	**17.2**	2.21		
20.1	1.80	**21.4**	1.63	22.7	1.48	**24.0**	1.33
20.2	1.78	**21.5**	1.62	**22.8**	1.46	**24.1**	1.32
20.3	1.77	**21.6**	1.61	**22.9**	1.45	**24.2**	1.31
20.4	1.76	**21.7**	1.59	**23.0**	1.44	**24.3**	1.30
20.5	1.74	**21.8**	1.58	**23.1**	1.43	**24.4**	1.29
20.6	1.73	**21.9**	1.57	**23.2**	1.42	**24.5**	1.28
20.7	1.72	**22.0**	1.56	**23.3**	1.41	**24.6**	1.27
20.8	1.71	**22.1**	1.55	**23.4**	1.40	**24.7**	1.26
20.9	1.69	**22.2**	1.53	**23.5**	1.38	**24.8**	1.25
21.0	1.68	**22.3**	1.52	**23.6**	1.37	**24.9**	1.24
21.1	1.67	**22.4**	1.51	**23.7**	1.36	**25.0**	1.22
21.2	1.66	**22.5**	1.50	**23.8**	1.35		
21.3	1.64	**22.6**	1.49	**23.9**	1.34		

Los datos de la *Tabla 1* son graficados como lo muestra la Fig. 6 evidenciando la característica no lineal del sensor.

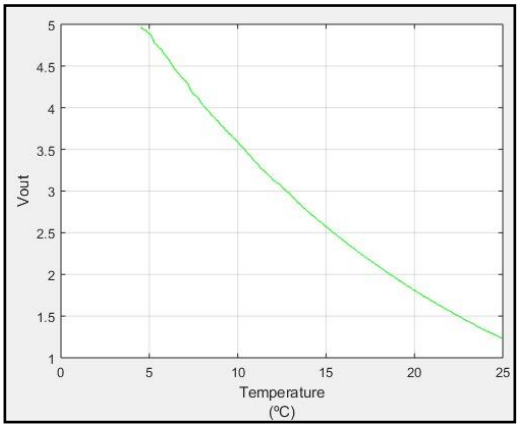

Fig. 6. Curva característica del termistor

2.3 Creación de la red neuronal

Se procede a crear la red neuronal e implementar el algoritmo de linealización en el sistema digital programable. La red neuronal propiamente no linealiza la respuesta del sensor, pero si logra crear un elemento de corrección que sumado a la señal de entrada retroalimentada logra compensar la no linealidad del sensor. Este método es resumido en el diagrama de la Fig. 7.

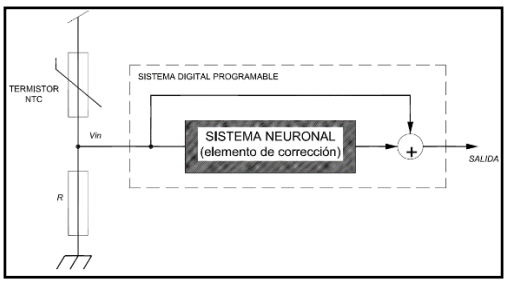

Fig. 7. Modelo de linealización para termistor

El propósito de la red neuronal, como ya se mencionó, es crear un elemento de corrección y para ello se requieren dos grupos de datos, datos de entrada y datos objetivo: los valores de tensión

de la salida no lineal del sensor obtenida del circuito de acondicionamiento (*Tabla 1*) se usarán como datos de entrada y otro grupo que se obtiene de las diferencias algebraicas entre las tensiones (*Tabla 1*) y la respuesta lineal deseada se usarán como datos objetivo.

Para la configuración inicial del código ANN en Matlab, se crean dos vectores: el primero es *in* el cual es un vector de todos los voltajes de salida de la aplicación ADC para el termistor, y el otro vector es *out* el cual son los datos de voltaje lineal menos voltajes de salida de la aplicación ADC para el termistor, después de esto se crea la red neuronal por medio del comando *net = newff*; agregado a esto se procede a mostrarle a la red que su entrada y salida son los vectores anteriormente creados y que tendrá dos neuronas en su primera capa.

El comando *trainlm* es una función de entrenamiento en red que actualiza los valores de peso y bias de acuerdo con la optimización de Levenberg-Marquardt. El comando *learngd* calcula el cambio de un peso dW para una neurona dada a partir de la entrada P de la neurona y el error E, y la tasa de aprendizaje de peso (o bias) LR, según el descenso del gradiente dw = lr * gW. El comando *mse* es una función de rendimiento de red, la cual mide el rendimiento de la red de acuerdo con la media de los errores al cuadrado. Para finalizar, el comando *dividerand* divide los objetivos en tres conjuntos usando índices aleatorios.

De este modo se crea la red neuronal y se establecen los parámetros básicos.

```
% 3. Configure the network
net.layers{1}.size = 2; %Hidden layer size
net.layers{1}.transferFcn = 'logsig'; %Hidden layer transfer function
%net.layers{1} %Hidden layer configuration
%net.inputWeights{1,1} %Properties of the weights of layer 1
net.layers{2}.size = 1; %Hidden layer size
net.layers{2}.transferFcn = 'tansig'; %Hidden layer transfer function
%net.layers{2} %Hidden layer configuration
%net.layerWeights{2,1} %Properties of the weights of layer 2
%net1 = configure(net,in,out);
%view(net)
```

Fig. 8. Configuración de la red en el código ANN en Matlab

En la configuración de la red neuronal en el código, se puede cambiar las neuronas que van a tener cada capa de la red neuronal, en este proyecto se utiliza una sola entrada, dos capas ocultas, donde la primera capa oculta tiene dos neuronas, la segunda solo una neurona, y una salida, es decir una red 1,2,1,1.

También se configura la función de activación que se utiliza para cada capa, utilizando la función sigmoidal para la primera capa, y para la segunda capa la función de tangente hiperbólica, debido a que en ese orden se obtienen mejores resultados de la red neuronal.

La función sigmoidal 'logsig' (Fig.9) posee como imagen un rango continuo de valores dentro de los intervalos [0,1] y viene dada por la ecuación (10).

$$F_1(x) = \frac{1}{1 + e^{-x}} \qquad (3)$$

La función tangente hiperbólica 'tansig'(Fig.9) posee como imagen un rango continuo de valores dentro de los intervalos [-1,1] y viene dada por la ecuación (11).

$$F_1(x) = \frac{2}{(1 + e^{-2x}) - 1} \qquad (4)$$

Ambas son funciones crecientes con dos niveles de saturación: el máximo que proporciona salida 1, y el mínimo salida 0 para la función sigmoidal y salida -1 para la función tangente hiperbólica.

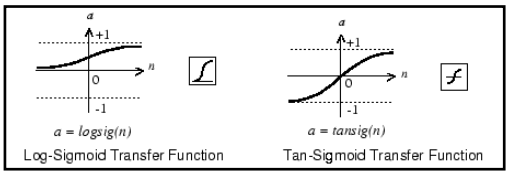

Fig.9. Funciones logsig y tansig dadas por Matlab

En el siguiente paso se inicializan las bias y los pesos (Fig.10), en el código se programan para que su valor inicial sea 1; esta configuración es de suma importancia debido a que el programa configura las bias y los pesos iniciales con valores genéricos muy cercanos a cero, con los cuales al finalizar el entrenamiento de la red neuronal se tenía problemas en la comprobación de resultados.

```
% 4. Initialize the weights and biases
net.iw{1,1}=ones(2,1);
net.lw{2,1}=ones(1,2);
net.b{1,1}=ones(2,1);
net.b{2,1}=ones(1,1);
```

Fig. 10. Inicialización de las bias y los pesos

En el entrenamiento de la red neuronal hay varios parámetros que se pueden modificar y todos estos son en función de cómo se requiere que la salida del entrenamiento sea: *init.net* carga la red con las configuraciones iniciales, bias, pesos, funciones de activación; *net.trainParam.epochs* configura la cantidad de iteraciones que se desea que la red sea entrenada para obtener convergencia, entre mayor sea el número hay más probabilidad de obtener mejores resultados, *net.trainParam.goal* es el error máximo permitido, *net.trainparam.max_fail* es el máximo número de fallas, *net.trainparam.min_grad* el mínimo rendimiento del gradiente, *net.trainparam.show* los resultados son visualizados siempre que transcurren este número de itareaciones, *net.trainparam.time* el tiempo utilizado para hacer el entrenamiento (Fig.11).

```
% 5. Train the network
net= init(net);
net.trainParam.epochs= 500; %Max no of epochs to train default=1000,
net.trainParam.goal= 0; %Performance goal default=0,
net.trainParam.max_fail= 6; %Max validation fail default=25,
net.trainParam.min_grad= 1e-7; %Max performance gradient default=1e-7;
net.trainParam.mu= 0.001; %Initail Mu default=0.001,
net.trainParam.mu_dec= 0.1; %mu decrease factor default=0.1,
net.trainParam.mu_inc= 10; %mu increase factor default=10,
net.trainParam.mu_max= 1e10; %Max mu default=1e10;
net.trainParam.show= 25; %Epochs b/w displays default=25,
net.trainParam.time= inf; %default
net = train(net,in,out);
y=net(in);
```

Fig. 11. Entrenamiento de la red

2.4 Implementación de algoritmo de linealización en el sistema digital programable

El sistema digital programable seleccionado es Arduino. En este se implementa y carga el algoritmo final que realiza todo el proceso, desde recibir la señal acondicionada del sensor hasta la impresión final del resultado en pantalla del LCD. Se busca principalmente implementar matemáticamente el comportamiento interno de la red neuronal, es decir, representar las funciones utilizadas en el entrenamiento neuronal que se realiza en Matlab. Para ello, se declaran los valores: w111 = 0.9636, w211 = 1.3688, w121 = -1.3603, w221 = 2.5334, b111 = -4.3996, b211 = -0.7362, b121 = -2.0340; estos valores corresponden respectivamente a los pesos y bias previamente obtenidos en Matlab y serán útiles para definir las funciones sigmoidal e hiperbólica descritas en (3 y **(4)**, definidas en el algoritmo, como se muestra en (12) y (13).

$$act\_1 = \frac{1}{(1 + exp(-(w111 * ADC\_Volt + b111)))} \quad (12)$$

$$act\_2 = \frac{1}{(1 + exp(-(w211 * ADC\_Volt + b211)))} \quad (13)$$

Para completar la representación matemática de la red se define su salida, así:

$$red = \frac{2}{(1 + exp(-2 * A))} - 1 \quad (14)$$

$$A = w121 * act\_1 + w221 * act\_2 + b121$$

Siguiendo la metodología planteada, es necesario sumar la salida de la red definida en **¡Error! No se encuentra el origen de la referencia.** al valor de la señal acondicionada del sensor y recibida también por el Arduino; logrando con ello corregir cualquier valor de la señal del sensor a un valor lineal.

$$v\_lin = ADC\_Volt + red \quad (15)$$

3. RESULTADOS

3.1 Respuesta linealizada del sensor

La validación de los datos se hace por medio del comando plot de la Fig.12, donde se toman los datos de temperatura por un lado y en el otro se suma la salida de la aplicación ADC para el termistor que es la entrada de la red neuronal con la salida de la red neuronal, al sumarse los datos deben ser lineales, como lo muestra la Fig.13.

```
% 6. Validate the network
Temp= 8:0.1:20;
plot(Temp, in + out)
xlabel({'Temperature','(°C)'})
ylabel({'Vout',})
```

Fig. 12. Validación de los datos

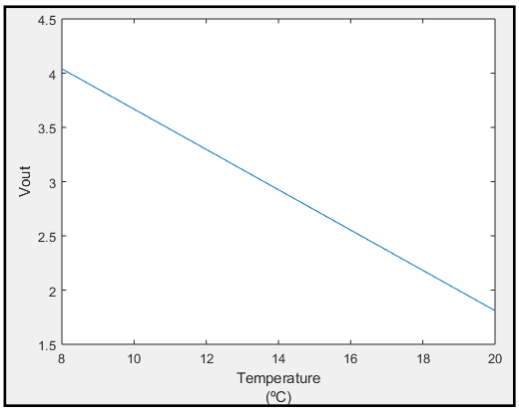

Fig. 13. Salida linealizada de la red neuronal

El código mostrado en la Fig.14 corrobora que por medio de las bias y pesos se puede obtener el algoritmo que utiliza Matlab para generar cada salida deseada con respecto a su entrada; *rta* es el valor de salida del algoritmo, *red* es la red entrenada en Matlab y *salida* es la salida de la red neuronal.

```
clc
i=10;
x= in(i)

bias1 = biases{1,1};
bias2 = biases{2,1};
weight.i = weights.i{1, 1};
weight.l = weights.l{2, 1};

b1 = bias1(1);
b2 = bias1(2);
b3 = bias2(1);
w1 = weight.i(1);
w2 = weight.i(2);
w3 = weight.l(1);
w4 = weight.l(2);

act1 = 1 / (1 + exp(-(w1*x+b1)));
act2 = 1 / (1 + exp(-(w2*x+b2)));

rta = 2/(1+exp(-2*(w3*act1+w4*act2+b3)))-1
Red=y(i)
Salida=out(i)
```

Fig.14. Código de verificación de valores

Una sencilla prueba se realiza como se muestra en la Fig.15, el valor de entrada a la red neuronal utilizado es 3.84 el cual está en el segmento de toma de datos de temperatura de 8° a 20° y la salida dada para este dato es 0.0330, al hacer el entrenamiento y pedir el dato de salida de la red es 0.0338 lo cual nos da un error de 2.42% para la cuarta cifra después del cero, lo que es un buen resultado para la conversión que se da en la programación del sistema digital programable Arduino.

También se logra por medio de ecuación alcanzar la misma salida de la red neuronal utilizando las bias y los pesos obtenidos por medio del entrenamiento de la red neuronal.

```
x =

    3.8400

rta =

    0.0388

Red =

    0.0388

Salida =

    0.0330
```

Fig. 15. Comprobación entrenamiento ANN

3.2 Prueba final del dispositivo desarrollado

Una vez realizado toda la implementación y desarrollo de la arquitectura física y lógica del dispositivo de medición propuesto en este documento, es necesario comprobar su funcionamiento; para ello se acude al laboratorio metrológico BioAsesores S.A.S. especializado en calibración de este tipo de equipos que se encuentra ubicado en la ciudad de Bucaramanga.

La metodología utilizada es por comparación directa de temperatura del dispositivo bajo prueba con respecto a las indicaciones del instrumento patrón, empleando un equipo "micro-baño de calibración" como generador o variador de temperatura. Se calibra para tres puntos de temperatura obteniendo los resultados de la Tabla 2.

La Taba 3 resume los porcentajes de error del dispositivo desarrollado, calculados.

Tabla 2: Resultados de calibración

Patrón	**Dispositivo bajo prueba**		
T [ºC]	T [ºC]	Error [ºC]	Incertidumbre ± [ºC]
11.000	11.3	0.26	0.15
13.996	14.2	0.20	0.26
16.990	17.3	0.31	0.26

Tabla 3: Resultados de calibración (% error)

Patrón	**Dispositivo bajo prueba**	
T [ºC]	T [ºC]	% Error
11.000	11.3	2.72
13.996	14.2	1.46
16.990	17.3	1.82

Como instrumento patrón se utiliza un termómetro digital del fabricante Fluke modelo 1523 y como instrumento generador un micro-baño de temperatura modelo 7103 del mismo fabricante. En la Fig. 16 se aprecian estos equipos.

Fig.16. Calibración de dispositivo

4. CONCLUSIONES

Se analiza, simula, e implementa el circuito de acondicionamiento para la aplicación ADC para el sensor de temperatura por medio del programa de Texas Instruments concluyendo que los valores obtenidos por medio de las ecuaciones son válidos para la obtención de una salida de voltaje que pueda recibir el conversor análogo/digital, dando a conocer que algunas de las resistencias son de un tamaño semejante a los obtenidos por ecuaciones debido a que no se encontraron con el mismo valor; pero eso no afectó o influyó significativamente ya que se contrastan los valores obtenidos en la salida del circuito de acondicionamiento con los de la simulación y son consecuentes.

Se analizaron diferentes parámetros para el entrenamiento de la red neuronal, siendo la más apropiada la configuración escogida para la implementación de este proyecto, debido a que la linealización de la curva es más precisa; y al verificar el error dado comparando la salida de la red y la salida deseada, se encuentra que por lo general son milésimas los cambios entre estos dos.

Las gráficas y resultados simulados fueron hechos por el software Matlab y evidencian el cumplimiento con una pequeña tasa de error cuando se compara con los datos de salida deseados a la salida de la red y los datos dados por la red neuronal entrenada.

La respuesta de la red neuronal implementada en forma matemática cumple su propósito en el sistema digital programable, dado que arroja valores que al ser comparados con la respuesta de la red entrenada en Matlab difieren mínimamente.

En la ejecución del sistema digital programable Arduino, se visualiza exactamente la temperatura con un bajo porcentaje de error respecto del instrumento patrón, y el voltaje ya linealizado.

REFERENCIAS

Quintero Camacho (2003). «*Método para linealizar la salida de un sensor*,» Ingeniería (Bogotá), pp. 82-85.

Rodríguez E. J. A., J. W. M. Ocampo y Ortega C. A. S. (2013). «*Medición de temperatura: sensores termoeléctricos*,» *Scientia et Technica*, p. 34

.

K. Souhil, D. Zohir y M. Fayçal, «Modeling of a smart humidity sensor,» *nternational Conference on Microelectronics*, pp. 135-138, 2008.

Naveen Kumar V. y Venkata Lakshmi N. K(2015) «*Investigation of Artificial Neural Network Techniques for Thermistor Linearization,» Sensors & Transducers Journal*, pp. 55-63.

Bishop J (2000) «*Thermistor Temperature Transducer to ADC Application,» Texas Instruments.*

Hudson M, (2014*). Neural Network Toolbox User´s Guide.*

Instruments T (2007). «*FAMILY OF 600µA/Ch 2.8MHz RAIL-TO-RAIL INPUT/OUTPUT*,»

Instruments T. (2015) «*REF200 Dual Current Source and Current Sink*,»

Medrano-Marque's (2000). «*A GENERAL METHOD FOR SENSOR LINEARIZATION BASED ON NEURAL*,» *ISCAS 2000 - IEEE International Symposium on Circuits and Systems*, pp. 1-3.

Sánchez A.M (2007). «*Instrumentación básica de procesos industriales*,» de *6851050 CURSO-ISA-PRESENTATION-INSTRUMENTACION-BASICA*.

VISHAY INTERTECHNOLOGY (2017), «*NTC Thermistors, Radial Leaded, Standard Precision*,».

BioAsesores S.A.S (2019). «C*ertificado de calibración número t63639-19*,» Bucaramanga.

Corzo Ruiz C. L(2017), «*Repositorio Institucional UTS - RI-UTS*,». [En línea]. Available: http://repositorio.uts.edu.co:8080/xmlui/bitstream/handle/123456789/364/BI-0375%20-%20Mora.pdf?sequence=1&isAllowed=y.

EETech L Media (2019).«*NTC thermistor » Resistor Guide,»* . [En línea]. Available: http://www.resistorguide.com/ntc-thermistor/.

Franco G. y Giraldo M. J. , «Biblioteca Digital Agronet,» (1998). [En línea]. Available: http://bibliotecadigital.agronet.gov.co/handle/11348/4039.

Ministerio de Agricultura y Desarrollo Rural, «Agronet,» (2014). [En línea]. Available: http://www.agronet.gov.co/Documents/Mora.pdf.