UNIVERSIDAD POLITÉCNICA SALESIANA

SEDE GUAYAQUIL

CARRERA DE INGENIERÍA ELECTRÓNICA

“Control de temperatura para un prototipo de sistema de avicultura mediante una red neuronal Adaline”

Trabajo de titulación previo a la obtención del

Título de Ingeniero Electrónico

AUTORES: Darwin Roberto Loachamin Flores
Félix Fidel Figueroa Garzón

TUTOR: Ing. Vicente Avelino Peñaranda

Guayaquil-Ecuador
2024

CERTIFICADO DE RESPONSABILIDAD Y AUTORÍA DEL TRABAJO DE TITULACIÓN

Nosotros, Darwin Roberto Loachamin Flores con documento de identificación N°0923657969 y Félix Fidel Figueroa Garzón con documento de identificación N°0926413238; manifestamos que:

Somos los autores y responsables del presente trabajo; y, autorizamos a que sinfines de lucro la Universidad Politécnica Salesiana pueda usar, difundir,reproducir o publicar de manera total o parcial el presente trabajo de titulación. Guayaquil,15 de febrero del año 2024

Darwin Roberto Loachamin Flores

0923657969

Félix Fidel Figueroa Garzón

0926413238

CERTIFICADO DE CESIÓN DE DERECHOS DE AUTOR DEL TRABAJO DE TITULACIÓN A LA UNIVERSIDAD POLITÉCNICA SALESIANA

Nosotros, Darwin Roberto Loachamin Flores con documento de identificación No. 0923657969 y Félix Fidel Figueroa Garzón con documento de identificación No. 0926413238, expresamos nuestra voluntad y por medio del presente documento cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del Proyecto técnico "Control de temperatura para un prototipo de sistema de avicultura mediante una red neuronal Adaline". , el cual ha sido desarrollado para optar por el título de: Ingeniero Electrónico, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En concordancia con lo manifestado, suscribimos este documento en el momento que hacemos la entrega del trabajo final en formato digital a la Biblioteca de la Universidad Politécnica Salesiana.

Guayaquil, 15 de febrero del año 2024

Atentamente,

Darwin Roberto Loachamin Flores

0923657969

Félix Fidel Figueroa Garzón

0926413238

CERTIFICADO DE DIRECCIÓN DEL TRABAJO DE TITULACIÓN

Yo, Vicente Avelino Peñaranda , con documento de identificación N°0916113426, docente de la Universidad Politécnica Salesiana, declaro que bajo mi tutoría fue desarrollado el trabajo de titulación: **"Control de temperatura para un prototipo de sistema de avicultura mediante una red neuronal Adaline"**, realizado por Darwin Roberto Loachamin Flores con documento de identificación No. 0923657969 y Félix Fidel Figueroa Garzón con documento de identificación No. 0926413238, obteniendo como resultado final el trabajo de titulación bajo la opción que cumple con todos los requisitos determinados por la Universidad Politécnica Salesiana.

1 de marzo del año 2024

Atentamente,

Ing. Vicente Avelino Peñaranda MSc.

DEDICATORIA

Dedico esta tesis a mis padres Jesús Amalia Flores y Roberto Loachamin, quienes con su apoyo y estuvieron pendientes de mí, a mis hermanas Jhoana Elizabeth Loachamin Flores y Blanca Almeida Flores que quiero mucho ya que todos siempre estuvieron apoyándome y nunca dejaron de creer en mí, me dieron todo el apoyo que necesitaba para poder llegar a ser un profesional de bien y que siempre tendrá mi más eterno agradecimiento.

Darwin Roberto Loachamin Flores

AGRADECIMIENTO

En primer lugar, doy gracias a mis padres que con sus cuidados fueron parte de mi desarrollo y despertaron en mí las ganas de aprender y analizar el porqué de las cosas. Me han demostrado el amor y me mostraron la ruta que debo seguir para ser una persona de bien como ellos lo son.

Agradezco a Grandes amigos que me dio la U porque en el proceso todos luchamos por llegar a la meta que es ser un profesional.

Agradezco a Andrea Huayamabe una de mis mejores amigas por siempre estar al pendiente de mí, aunque no estaba en la misma universidad.

Agradezco a mi novia Arantxa Sánchez, que llego en estos últimos momentos y siempre me ha impulsado a terminar este proceso y creyó en mis capacidades.

Me Agradezco a mí mismo por no tirar la toalla y seguir luchando hasta terminar este proceso después de muchas trabas y poder conseguir esta meta que es ser profesional.

Darwin Roberto Loachamin Flores

DEDICATORIA

Este proyecto de titulación va dedicado en memoria de mi padre fallecido Félix Fidel Figueroa Rivas, quién fue una persona sabia, trabajadora y luchadora quien me enseño los valores de la vida, a ser valiente, responsable y a enfrentar cada desafió con sabiduría, a mi madre Gina Lucía Garzón Ortiz quién con su amor y sus sabios consejos me han alentado a continuar, gracias a por su gran apoyo incondicional me han permitido cumplir hoy una de las metas más importante en mi vida, ellos fueron la razón fundamentalpor la que pude terminar mis estudios, gracias a su sacrifico y la fe que tuvieron en mí he llegado a donde estoy.

Félix Fidel Figueroa Garzón

AGRADECIMIENTO

Agradezco principalmente a Dios por ser el pilar fundamental en mi vida, quién me ha guiado y dado la fortaleza para seguir adelante, por permitirme llegar hasta este momento tan especial.

Agradezco a mis padres por haberme brindado la oportunidad de estudiar y ser ese motor primordial que me inspira a ser mejor cada día, de estar apoyándome en todo momento en cada paso importante de mi vida.

Agradezco a mi compañero y amigo de tesis Darwin Loachamin por todo el compromiso y la buena comunicación contribuimos a terminar el trabajo con éxito.

Agradezco al Ing. Andrés Pilco quien fue mi compañero y gran amigo de estudio por ser ese apoyo permanente en cada trabajo grupal ya sea de deberes, talleres y proyectos.

Agradezco a todos mis amigos que me brindó la Universidad en este proceso de formación profesional.

Félix Fidel Figueroa Garzón

RESUMEN

AÑO	ALUMNOS	DIRECTOR DE PROYECTO TÉCNICO	TEMA DE PROYECTO TÉCNICO
2024	Darwin Roberto Loachamin Flores Félix Fidel Figueroa Garzón	Ing. Vicente Avelino Peñaranda, MSc.	Control de temperatura para un prototipo de sistema de avicultura mediante una red neuronal Adaline

El presente proyecto de titulación tiene como objetivo el control de temperatura para un prototipo de sistema de avicultura mediante una red neuronal Adaline. La constante necesidad del consumo de aves hace que se busque nuevas formas de control del ambiente en donde se desarrolla su crecimiento.

La incorporación de microcontroladores como es el caso del ESP32 y la utilización de redes neuronales, facilita el desarrollo de un control adecuado capaz de mantener un galpón de criadero de pollos con una temperatura acorde al tiempo de vida de las crías.

Se diseñó e implementó una maqueta que simula un galpón para el criadero de pollos en su primera semana de vida, en la cual se controla la temperatura con la ayuda de sensores adecuadamente instalados. Adicionalmente se puede medir la humedad del ambiente y se genera una circulación de aire con la ayuda de un ventilador.

En el proyecto el entrenamiento se refiere a tomar los cuatro valores de temperatura a través de los sensores, y con la red Adaline se realiza un muestreo general con lo cual se genera una tabla de eventos dependiendo de una referencia estable lo que produce una salida que acciona un calentador según

los valores de +1 o -1.

Para el monitoreo de las temperaturas, humedad, ventilación y encendido del calentador se utilizó internet de las cosas (IoT) del tipo Ubidots el cual interactúa con el ESP32.

Palabras claves: Microcontrolador, ESP 32, Red Adaline, temperatura,humedad, sensores, IoT, Ubidots.

ABSTRACT

YEAR	STUDENTS	DIRECTOR OF TECHNICAL PROJECT	TECHNICAL PROJECT THEME
2024	Darwin Roberto Loachamin Flores Félix Fidel Figueroa Garzón	Ing. Vicente Avelino Peñaranda, MSc.	Temperature control for a prototype poultry farming system using an Adaline neural network

The objective of this degree project is temperature control for a prototype poultry farming system using an Adaline neural network.
The constant need to consume birds makes us look for new ways to control the environment where their growth takes place.

The incorporation of microcontrollers such as the ESP32 and the use of neural networks facilitates the development of adequate control capable of maintaining a chicken breeding shed with a temperature according to the life span of the offspring.

A model was designed and implemented that simulates a shed for the chicken farm in its first week of life, in which the temperature is controlled with the help of properly installed sensors. Additionally, the humidity of the environment can be measured and air circulation is generated with the help of a fan.

In the project, the training refers to taking the four temperature values through the

sensors, and with the Adaline network a general sampling is carried out, which will generate a table of events depending on a stable reference, which generates an output that activates a heater based on the values of +1 or -1.

To monitor temperatures, humidity, ventilation and turning on the heater, the Internet of Things (IoT) of the Ubidots type was used, which interacts with the ESP32.

Keywords: Microcontroller, ESP 32, Adaline Network, temperature, humidity, sensors. IoT, Ubidots.

INDICE GENERAL

Contenido

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Las granjas avícolas requieren energía para mantener condiciones térmicas óptimas y niveles adecuados de luminosidad durante todas las etapas de crecimiento y desarrollo de las aves, lo cual está influenciado por el clima de la región. (Cabrera, 2017).

La iluminación juega un papel crucial en la calidad del producto avícola, su manejo y el comportamiento de las aves en sistemas intensivos, lo que puede afectar el peso final de las aves al término de la producción. En los últimos años, el avance tecnológico de las redes inalámbricas ha experimentado un rápido desarrollo, lo que facilita la implementación de redes sensoriales inalámbricas (WSN) con módulos microcontroladores. Esto se debe a sus costos accesibles, bajo mantenimiento y consumo reducido de energía. Por esta razón, este tipo de redes se aplican comúnmente en seguridad en centros comerciales, áreas propensas a desastres naturales (como desbordamientos de ríos o sismos), control de procesos industriales, seguimiento de pacientes en hospitales e incluso en sectores agrícolas. (Cabrera, 2017).

La investigación que está desarrollada en este trabajo se basa en realizar un diseño de red formada de nodos compuestos por sensores de temperatura, módulo microcontrolador ESP32, con sus características y la utilización de la red Adaline. (Cabrera, 2017).

En la primera parte de este trabajo de titulación se refiere a el problema el cuál se toma como información para el planteamiento de este, como son los antecedentes, el alcance, objetivo principal y específicos del proyecto.

La segunda sección, se refiere a los fundamentos teóricos, en donde se explica detalles y características de los sistemas que utilizan redes neuronales, su clasificación los dispositivos utilizados en el proyecto con sus respectivas

características. También se describe las características del microcontrolador ESP32 con sus diversas posibilidades de aplicaciones. Se explica sobre el funcionamiento de la plataforma IoT y sus principales características.

En la tercera sección, se detalla la implementación del sistema de control de temperatura del galpón, las conexiones entre los distintos sensores de temperatura, humedad, calefacción, Se describen algunos algoritmos según el procesamiento de la variable correspondiente.

En la cuarta sección, se describe las pruebas realizadas del control de temperatura como a respuesta de lectura que se obtiene al aplicarle una perturbación con el ventilador. Se puede observar los valores digitales y análogos de la humedad, temperatura en los cuatro puntos, accionamiento de las salidas de calefacción y ventilación utilizando el IoT Ubidots.

Así también se puede ver todas las variables en forma remota como una Tablet o en un teléfono inteligente con la ayuda del Ubidots.

1. EL PROBLEMA

1.1. Antecedentes

Una parte de la familia Loachamin se dedica al criadero de pollos, pero han tenido problemas en mayor parte con los que tienen la primera semana de vida y consideraron que una de las posibles causas es por la falta de un controlde temperatura adecuado en el ambiente.

La falta de un sistema que mantenga una temperatura ambiente adecuada en el galpón y una ventilación adecuada, son factores que pueden atribuir a la muerte prematura de cierta cantidad de pollos en su primera semana de vida.

Como se quería primero realizar pruebas de un controlador de temperatura, se realizó una maqueta la cual tiene cuatro puntos de medición de temperatura, ventilación forzada y un calentador formado por un led de potencia.

1.2. Importancia y Alcances

La implementación y equipamiento de tecnologías integradas como ésta, permite a las haciendas crear un ambiente acondicionado para las aves, mejorando de esta manera su productividad y por supuesto incrementando losingresos en las ventas. (Cabrera, 2017).

La red con sensores para el control de parámetros ambientales ayuda a mejorar el desarrollo y calidad de las aves y por consiguiente la producción teniendo un sistema de control en donde se puedan adecuar los parámetros zootécnicos de las aves llegando también a producirlas a mayor escala y llevar un mejor control con una mayor cantidad de aves. Se mejora de manera directa la calidad y eficiencia en la producción debido a la poca

afección de los parámetros climáticos en el desarrollo de las aves. (Cabrera, 2017).

Para la realización de este proyecto se utilizó una interfaz representada por un microcontrolador ESP32, la cual coge las señales de los sensores y con laayuda de la red neuronal Adaline, procesa las mismas y se emite las señales de salida para el calentador representado por un led de potencia. El criadero de pollos esta complementado como una maqueta la cual tiene instalado 4 sensores de temperatura adecuadamente ubicados y conectados con elESP32 para sacar un promedio y posteriormente dar un entrenamiento a la red Adaline. Con esto se emitirá salidas de -1 cuando se deba calentar y +1 cuando se deje de calentar por medio del foco led de potencia. En todo momento se está midiendo la humedad del ambiente y con la ayuda de un ventilador se puede refrescar el sector. La parte del control se la realiza cuando los pollos están en su primera semana de nacidos, ya que es cuandose requiere un mejor control de temperatura. Todas las variables son monitoreadas con la ayuda de un IoT Ubidots.

1.3. Delimitación

1.3.1 Temporal

Este trabajo de grado se lo realizó entre noviembre del 2023 y febrero del 2024.

1.3.2 Espacial

El proyecto técnico, está diseñado y ubicado en el hogar de la familia Loachamin.

1.3.3 Académica

El propósito del presente proyecto de grado consiste en desarrollar y poner en práctica un módulo de capacitación empleando el procesador ESP32, dirigido a aplicaciones centradas en el ámbito de la Domótica.

1.4. Innovación

El desarrollo de este trabajo de grado de un sistema neuronal tipo Adaline tiene un impacto innovador ya que normalmente se utiliza controles tradicionales como es el PID, difuso, etc. En este caso se utiliza un control coninteligencia artificial en cual realiza un entrenamiento de los diferentes puntosde medición en el galpón y según el setpoint activa o desactiva un calentador para mantener los grados centígrados adecuados para los pollos en su primera semana de vida.

1.5. Objetivos

1.5.1 Objetivo general

Implementar un prototipo para el control y monitoreo de temperatura utilizando una red neuronal Adaline para un sistema de avicultura.

1.5.2 Objetivos específicos

- Realizar el programa de la red neuronal Adaline para el control y monitoreo de temperatura.

- Diseñar un plano esquemático utilizando sensores, actuadores, y un control de temperatura que contenga al microcontrolador ESP32.

- Realizar las conexiones de sensores, actuadores e interfaces para el control de temperatura del criadero de pollos.

- Presentación de los resultados en diferentes graficas de las variables físicas del control neuronal, tanto en el PC como a distancia utilizando internet de las cosas (IoT).

2. FUNDAMENTOS TEÓRICOS

2.1. La Avicultura en el Ecuador

En el Ecuador, la explotación avícola se da en las tres regiones: Costa, Sierra, Oriente, excepto en la región Insular, y es el pollo una de las carnes más utilizadas para la alimentación en el país. El ingreso y el éxito en una empresa avícola están en relación directa con la capacidad y pericia de quien lo establece y administra; el avicultor debe consagrarse por entero al negocio. La avicultura es una de las cadenas de mayor importancia del sector agropecuario ecuatoriano, por el aporte a la seguridad alimentaria del pueblo, generación de empleos, además de los ingresos para pequeños productores del maíz y soya, que son las materias primas más utilizadas en la alimentación de las aves. (Erazo & Salgado, 2014).

2.1.1 Construcción y Adecuamiento del Galpón

El galpón ideal se orienta de este a oeste para evitar que la luz solar penetre en el interior durante el amanecer y el atardecer, lo que previene el hacinamiento de los pollitos en la sombra y reduce el riesgo de mortalidad. En áreas con corrientes de aire fuertes, es importante establecer barreras naturales, como árboles, o artificiales, como cortinas de polímero, para cubrir las entradas del galpón y mantener una temperatura estable.

Las dimensiones del galpón dependen del número de pollitos que se pretenda alojar y al entorno donde se encuentre el criadero. (Erazo & Salgado, 2014).

En la siguiente tabla 1 se muestra la cantidad de aves por metros en el galpón.

CLIMA	CANTIDAD DE AVES X METRO CUADRADO
Frio	10 pollos x metro cuadrado
Templado	9 pollos x metro cuadrado
Cálido	8 pollos x metro cuadrado

Tabla 1: Cantidad de aves en un galpón de acuerdo con el clima *(Erazo & Salgado, 2014).*

De acuerdo con la tabla 1 pode deducir que las dimensiones de un galpón que albergara 15000 pollos en clima Frio (15000/10) debería tener una extensiónde 1500 metros cuadrados. El ancho aconsejable en la construcción de galponeses de 10 a 15 metros, por lo que el largo del galpón a calcular con un ancho de 15 metros (15000/15) sería de 100 metros. (Erazo & Salgado, 2014).

2.1.2 Factores Ambientales

Temperatura

Durante el proceso de crianza de los pollitos, es fundamental mantener niveles de temperatura adecuados, especialmente durante las dos primeras semanas de vida. Durante este período, los pollitos no tienen la capacidad de regular sus procesos metabólicos ni controlar eficazmente su temperatura corporal, por lo que dependen completamente de la temperatura del entorno para mantenerse en condiciones óptimas. Los pollitos desarrollan su capacidad de regular la temperatura alrededor de los 12 a14 días de edad. (Erazo & Salgado, 2014).

A continuación, se muestra en la tabla 2 la temperatura que pueden tener las crías según los días.

EDAD	Temperatura (°C)
1o.-2o.Día	32-33
3o.-7o. Día	29-30
2a. Semana	27-29
3a. Semana	25-27
4a. Semana	23-25
5a. Semana en Adelante	21-23

Tabla 2: Temperatura de crianza (Erazo & Salgado, 2014).

Si la temperatura ambiental varía, también lo hará la temperatura corporal del pollito. Así mismo si aumenta la temperatura corporal de los pollitos también aumenta la temperatura ambiental interna del criadero. (Erazo & Salgado, 2014). Al tener cambios bruscos de temperatura es decir demasiado frio o calor durante las primeras semanas, se puede ocasionar una mala conversión alimenticia o mayor susceptibilidad a enfermedades e incluso la muerte de los pollitos. La zona de neutralidad térmica de las aves se encuentra entre 15 y 25 °C, al disminuir o exceder esta zona el ave empieza a presentar determinados problemas en su organismo. (Erazo & Salgado, 2014).

Humedad

La capacidad del aire para mantener la humedad depende de su temperatura. El aire tibio puede contener más humedad que el aire frío. El término humedad relativa se refiere al porcentaje de saturación de agua en el aire a cualquier temperatura dada. El nivel de humedad influye en la capacidad del ave para enfriarse mediante el jadeo, e influye en la producción de amoníaco. (Erazo & Salgado, 2014).

El cuerpo del ave está compuesto aproximadamente por un 70% de agua, resultado de su consumo de dos a tres litros de agua por kilogramo de alimento. Gran parte del agua ingerida por el ave regresa a la caseta de producción a través de las deposiciones, lo que aumenta la humedad del ambiente. La gallinaza producida por cada ave contiene alrededor del 70% de humedad; un pollo de carne de siete semanas elimina aproximadamente 5 kg de gallinaza. Los niveles de humedad relativa dentro de la caseta varían con una temperatura baja. A temperatura menor, mayor será la humedad y a temperatura mayor menor será la humedad. (Erazo & Salgado, 2014).

En la tabla 3, se muestra la humedad requerida en un proceso Avícola.

EDAD	Humedad R. (%)
1o.-2o.Día	50-55
3o.-7o. Día	50-60
2a. Semana	55-60
3a. Semana	60-65
4a. Semana	65-70
5a. Semana en Adelante	65-70

Tabla 3: Humedad requerida en el proceso Avícola.

Según la información proporcionada en la tabla, es recomendable supervisar diariamente el nivel de humedad relativa en el gallinero. Si este desciende por debajo del 50% durante el proceso de producción, el ambiente tiende a volverse seco y polvoriento, lo que puede provocar deshidratación y problemas respiratorios en las aves, afectando así negativamente la producción. Por el contrario, si los índices exceden el 70% de humedad en la cama se tiende a aumentar los malos olores (Amoniaco) y provoca la proliferación de enfermedades y hongos. (Erazo & Salgado, 2014).

Ventilación

La principal manera de controlar el ambiente de las aves es manejando la ventilación, pues es esencial aportar aire de buena calidad en forma constante y uniforme al nivel de las aves. En todas las etapas del crecimiento, los pollos necesitan aire fresco para conservar la salud y lograr todo su potencial. (Erazo & Salgado, 2014).

La ventilación ayuda a mantener las temperaturas dentro del galpón, dentro dela “zona de confort” de los animales. Durante las primeras etapas del período de producción la principal preocupación es mantener a las aves con el calor suficiente, pero conforme crecen, el principal objetivo es mantenerlas suficientemente frescas. Los galpones y los sistemas de ventilación que se utilicen dependen del clima, pero en todos los casos la ventilación efectiva debe

eliminar el exceso de calor y humedad, proporcionar oxígeno y mejorar la calidad del aire al eliminar los gases nocivos. (Erazo & Salgado, 2014).

2.2. Redes Neuronales

La Agencia de Investigación de Proyectos Avanzados de Defensa, define una red neuronal artificial como un sistema compuesto de muchos elementos simples de procesamiento los cuales operan en paralelo y cuya función es determinada por la estructura de la red y el peso de las conexiones, donde el procesamiento se realiza en cada uno de los nodos o elementos de cómputo. (Rodríguez, 2020).

Las redes neuronales imitan la estructura y función del cerebro humano al procesar información de manera no lineal y realizar cálculos en paralelo a gran escala, con una capacidad de almacenamiento distribuido. Al enfrentarse a diversos tipos de información, estas redes aprenden de manera autónoma y ajustan continuamente sus sistemas para adaptarse a entornos variables.
Por lo tanto, se usan ampliamente en el reconocimiento de patrones, para predecir cambios en las cosas, optimizar decisiones, mejorar el control del proceso y otrastareas. (Rodríguez, 2020).
Una red neuronal es un procesador paralelo masivamente distribuido que posee una capacidad innata para almacenar conocimiento adquirido a partir de la experiencia, para luego hacerlo utilizable. Se asemeja al cerebro en dos aspectos fundamentales:

- La adquisición de conocimiento por parte de la red se lleva a cabo mediante un proceso de aprendizaje.
- Las conexiones entre neuronas, conocidas como pesos sinápticos, con utilizadas para almacenar dicho conocimiento. (Rodríguez, 2020).

El algoritmo Levenberg-Marquardt se utiliza en redes neuronales como un método de entrenamiento no lineal. Este algoritmo combina el descenso de gradiente con el método Quasi-Newton para garantizar una rápida convergencia local y mantener un rendimiento general óptimo. Su concepto fundamental radica en asegurar que cada iteración no sobrepase lo necesario a lo largo de un solo

gradiente en dirección negativa, lo que permite la búsqueda del error en la dirección de descenso.

Además, los pesos de la red pueden optimizarse mediante el ajuste adaptativo entre el método de descenso de gradiente más pronunciado y el método de Gauss-Newton, que permite que la red converja de manera efectiva y esto mejora en gran medida la velocidad de convergencia y la generalización de la red. (Rodríguez, 2020).

Las redes neuronales pueden emplearse para desarrollar controladores altamente no lineales, donde los pesos o parámetros internos se determinan automáticamente mediante un proceso de aprendizaje. Modificar el peso de un elemento afectará el comportamiento de la red, lo que permite adaptar su funcionamiento según las necesidades específicas de la aplicación. El objetivo es encontrar los pesos de la red para lograr una relación entrada/ salida deseada. (Rodríguez, 2020).

La superioridad de las redes neuronales artificiales sobre los sistemas matemáticos o expertos radica en que su funcionalidad se vuelve más compleja a medida que aumenta el número y la variedad de neuronas involucradas. Similar al funcionamiento de las neuronas biológicas, la pérdida o el deterioro de una sola neurona afecta cuantitativamente pero no cualitativamente al sistema en su conjunto. Esto les confiere características que las hacen muy adecuadas para la realización de tareas tales como identificación, reconocimiento de patrones y sobre todo control. (Rodríguez, 2020).

La organización de las neuronas en una red neuronal artificial se lleva a cabo mediante la formación de capas compuestas por un número específico de neuronas. Cuando un grupo de neuronas artificiales recibe información similar simultáneamente, se le denomina capa. En una red pode diferenciar tres tipos de niveles: (Rodríguez, 2020).

• Entrada: Es el conjunto de neuronas que recibe directamente la información proveniente de las fuentes externas de la red.

• Oculto: Corresponde a un conjunto de neuronas internas a la red y no tiene contacto directo con el exterior.

• Salida: Es el conjunto de neuronas que transfieren la información que la red ha procesado hacia el exterior. La arquitectura de una red neuronal artificial ver figura 1, es la forma como se organizan las neuronas en su interior y está estrechamente ligada al algoritmo de aprendizaje usado para entrenar la red. (Rodríguez, 2020).

Dependiendo del número de capas, definí las redes como monocapa y multicapa.

Una red neuronal artificial monocapa es capaz de aprender por sí misma para controlar sistemas dinámicos no lineales, es decir, que sus parámetros cambian conforme pasa el tiempo, esto es quizá una de las razones por las que más se utiliza este tipo de control. (Rodríguez, 2020).

Una red neuronal de varias capas aprende a reconocer las características dinámicas del sistema. Por otro lado, un controlador, también una red neuronal de múltiples capas, aprende a regular el emulador. Este controlador autoaprendizaje se emplea para supervisar el sistema dinámico real. El proceso de aprendizaje continúa a medida que el emulador y el controlador mejoran y rastrean el proceso físico el cual es el cálculo procedente al relacionar las entradas y salidas del sistema. (Rodríguez, 2020). En la figura 1 se muestra la estructura de una red neuronal.

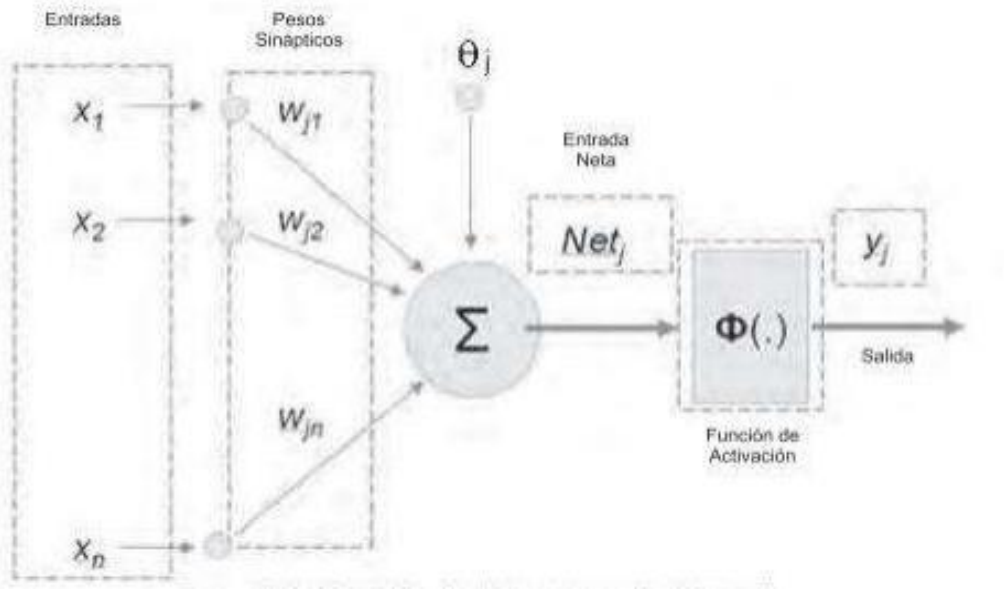


Figura 1: Estructura de una red neuronal artificial *(Rodríguez, 2020).*

Las redes o sistemas neuronales artificiales constituyen en la actualidad un activo campo multidisciplinario, en el que confluyen investigadores procedentes de muy diferentes áreas, como la electrónica, física, matemáticas, ingeniería, biología o psicología. Sin embargo, es fundamental tener en cuenta que en las redes neuronales artificiales no se busca alcanzar ningún tipo de ambición prometeica, como en los primeros días de la IA. En cambio, se emplean para el control de procesos industriales, la identificación de vehículos en peajes de autopistas o la predicción del consumo eléctrico. Estos son objetivos mucho más modestos que la creación de un cerebro artificial, pero son extremadamente útiles desde una perspectiva tecnológica.

Es en este tipo de problemas prácticos en los que los sistemas neuronales están alcanzando excelentes resultados. (Rodríguez, 2020).

2.2.1 El Perceptrón

El Perceptrón fue el pionero entre los modelos de redes neuronales artificiales, presentado a la comunidad científica por el psicólogo Frank Rosenblatt en 1958. Su introducción generó un gran interés en la década de los años sesenta debido a su habilidad para aprender a reconocer patrones simples con una superficie de separación lineal. Sin embargo, también atrajo críticas severas que eventualmente llevaron a que la propuesta de Rosenblatt cayera en el olvido. La estructura del Perceptrón es supremamente sencilla: en su entrada posee varias neuronas lineales que se encargan de recibir el estímulo externo de la red y a la salida presenta una única capa de neuronas, con función de activación binaria o bipolar lo cual trae como consecuencia que la salida de cada neurona esté entre dos posibles valores. (Rodríguez, 2020).

La estructura del perceptrón que se aprecia en la figura 2, se inspira en las primeras etapas de procesamiento de los sistemas sensoriales de los animales, por ejemplo, el de visión, en los cuales la información va atravesando sucesivas capas de neuronas, que realizan un procesamiento progresivamente de más alto nivel. El perceptrón simple es un modelo unidireccional, compuesto por dos capas de neuronas, una sensorial o de entradas, y otra de salida. (Rodríguez, 2020).

Figura 2: Perceptrón simple y función de transferencia de su neurona. (Rodríguez, 2020).

Las neuronas de entrada no realizan ningún computo, únicamente envían la información a las neuronas de salida. La función de activación de las neuronas de la capa de salida es de tipo escalón o Heaviside. (Rodríguez, 2020).

El perceptrón puede desempeñar roles tanto como clasificador como en la representación de funciones booleanas. Esto se debe a que su neurona es esencialmente de tipo MacCulloch-Pitts, con una salida binaria. Cada neurona del perceptrón representauna determinada clase, de modo que, dado un vector de entrada, una cierta neurona responde con 0 si no pertenece a la clase que representa, y con un 1 sipertenece. (Rodríguez, 2020).

Es evidente que una neurona tipo perceptrón solo puede discriminar entre dos clases que sean linealmente separables. Por lo tanto, a pesar de su considerable interés inicial, el perceptrón tiene limitaciones significativas, ya que solo puede representar funciones que sean linealmente separables. Así, aunque pueda aprender automáticamente a representar complejas funciones booleanas o resolver con éxito muchos problemas de clasificación, en otras ocasiones fallará estrepitosamente. (Rodríguez, 2020).

2.2.2 Red Adaline

Casi simultáneamente al desarrollo del Perceptrón, Bernard Widrow y su estudiante Marcian Hoff presentaron su red neuronal ADALINE. Esta red es similar al Perceptrón, pero en lugar de tener una función de activación binaria o bipolar, posee una función de activación lineal. La red ADALINE, a pesar de tener

las mismas limitaciones de la red Perceptrón, fue un gran adelanto en el desarrollo de las redes neuronales pues el mecanismo de aprendizaje que utiliza es basado en elementos de la teoría de optimización; específicamente se usa la propuesta del gradiente descendente para la deducción de la regla de entrenamiento. (Rodríguez, 2020).

El uso del gradiente descendente es la base de los algoritmos de aprendizaje que hoy se usan para entrenar redes neuronales más sofisticadas. Uno de los aspectos más interesante de la red ADALINE es su aplicación en problemas de filtrado de señales, pues su estructura se ajusta a la de un filtro adaptativo. (Rodríguez, 2020).

Es importante recalcar que la red tipo ADALINE puede tener más de una neurona en su capa de procesamiento, dando origen a un sistema con N entradas y M salidas. La estructura de la red ADALINE la cual se aprecia en la figura 6, es similar a la del perceptrón con la única diferencia de que la red ADALINE posee una función de activación lineal. No obstante, la diferencia más importante con el perceptrón y red Adaline reside en la regla de aprendizaje que implementa. (Rodríguez, 2020).

En la adalina se utiliza la regla de Widrow-Hoff, también conocida como regla LMS (Least Mean Squares, mínimos cuadrados), que conduce a actualizaciones de tipo continuo, siendo la actualización de los pesos proporcional al error que la neurona comete. (Rodríguez, 2020).
A continuación, se observar en la figura 3 la estructura de una red neuronal del tipo Adaline.

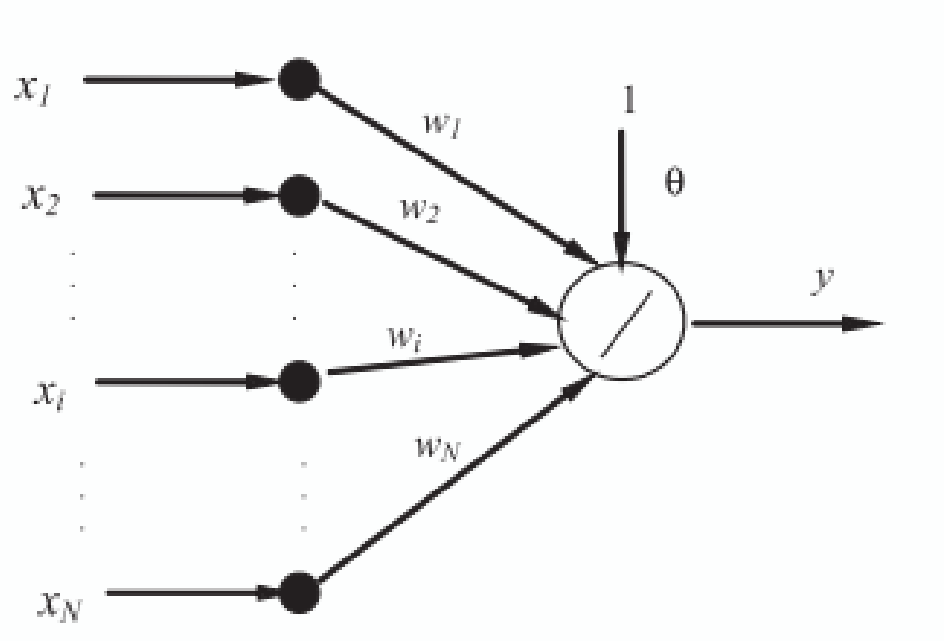


Figura 3: Estructura de una red neuronal tipo Adaline *(Rodríguez, 2020)*.

La Adalina se viene utilizando con asiduidad desde los años sesenta como filtro adaptativo, por ejemplo, para cancelar el ruido en la transmisión de señales. Un ejemplo clásico es su empleo como supresor de ecos en las comunicaciones telefónicas por satélite. (Rodríguez, 2020).

Debido a su naturaleza lineal, la utilidad del ADALINE se ve limitada. Por lo tanto, solo puede separar correctamente patrones que sean linealmente independientes, y en ocasiones falla al intentar discriminar patrones que sean linealmente separables, los cuales el perceptrón puede manejar con éxito. No obstante, ante patrones no separables linealmente, los resultados que proporciona son en promedio mejores que los del perceptrón pues la Adalina siempre opera reduciendo el error cuadrático medio al mínimo posible. (Rodríguez, 2020).

2.3 Ubidots

Ubidots capacita a entusiastas de la tecnología y emprendedores para desarrollar nuevos productos y aplicaciones basados en objetos conectados a Internet. Esto abarca desde la creación de una aplicación que registre la ingesta de cafeína a lo largo del día y envíe alertas por mensaje de texto si se excede

cierto nivel, hasta el monitoreo en tiempo real de la temperatura de una nevera industrial desde la Nube.

Ubidots permite crear aplicaciones para el “IoT” en poco tiempo y su interfaz es amigable con el usuario. (Espinosa & Orellana, 2021).

Es un servicio gratuito que permite a los desarrolladores crear un entorno de computación en la nube asequible, fiable y utilizable en un ecosistema de plataformas “IoT”. Ubidots está especializado en soluciones de hardware y software conectadas para monitorizar, controlar y automatizar procesos remotamente en el ámbito de la salud, energía, industria, fabricación, servicios públicos y transporte. (Espinosa & Orellana, 2021).

Con el fin de crear aplicaciones desde básicas hasta avanzadas. Los principales beneficios de la plataforma Ubidots son: (Espinosa & Orellana, 2021).

- La conexión del hardware con la nube y diversas bibliotecas.
- Configuración de variables de forma automática, configurable sus propiedades y su apariencia en los dispositivos para duplicar el proceso en nuevos dispositivos.
- Modificación de la API (Application Programming Interface).
- Supervisión para el análisis de datos de las aplicaciones con integraciones API.
- Conversión de datos de origen nativos en información con variables sintéticas.
- Diseño para cuadros de mando analizados en tiempo realpara el análisis de datos y control en los dispositivos.
- Mejora la función compartir datos con enlaces públicos o integrando cuadros de mando o widgets en aplicaciones web privadas y móviles.
- Los siguientes comandos "kill switch" o "restart" se activan cuando el hardware ha estado inactivo durante mucho tiempo.
- Asignación de permisos y restricciones para cualquier usuario que busque interactúe con los mandos, dispositivosy/o eventos. (Espinosa & Orellana, 2021)

Ubidots maneja 4 términos necesarios para el manejo de datos en la nube: Data Source,Variable, Value y Event. A continuación, se detallacada uno de estos: (Espinosa & Orellana, 2021).

- Data Source: maneja un conjunto de información haciendo referencia a un dispositivo, en donde cada data source utiliza una o más variables.
- Variable: maneja un conjunto de datos que varía con el tiempo.
- Value: es un valor actual de una variable en un momento determinado.
- Event: permite tomar una acción en un instante determinado. Permite la interacción con cada elemento de ubidots a través de su API, por lo tanto, cada elemento que presenta puede ser creado, modificado o eliminado. (Espinosa & Orellana, 2021)

Figura 4: Logo Ubidots. (Espinosa & Orellana, 2021)

2.3.1 Creación de una cuenta

Para ser un usuario en la nube de Ubidots se debe iniciar con un usuario en su servidor, con un email y una contraseña. Esto se realiza desde su página principal: *"https://ubidots.com"*. (Espinosa & Orellana, 2021).

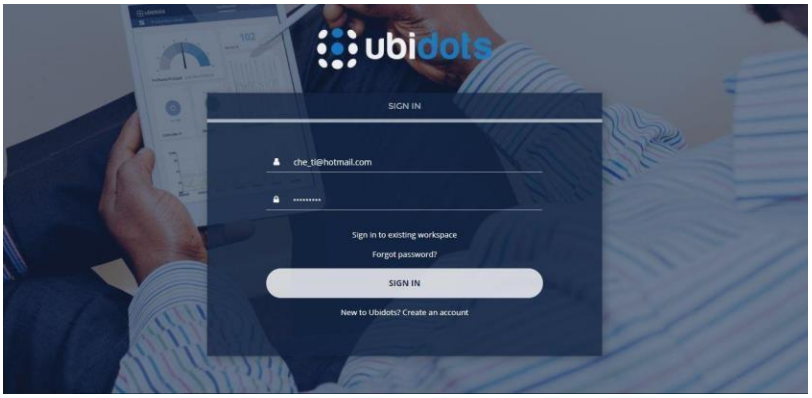


Figura 5: Interfaz de inscripción en Ubidots.

2.3.2 Dispositivos.

Los dispositivos contienen datos almacenados en variables, para acceder a un dispositivo es necesario contar con el token o key API; loque permite la lectura y modificación de las variables almacenadas. (Espinosa & Orellana, 2021).

2.3.3 Variables

Ubidots almacena los datos en variables, estas variables pueden ser: sintéticas o cruda.

Las variables crudas son variables extraídas desde un ambiente externo como Node red donde los datos llegan sin ser procesados, mientras que las variables sintéticas, son variables con las cuales se ha realizado cálculos dentro de Ubidots por lo tanto son variables que modifican su valor respecto a un cálculo. (Espinosa & Orellana, 2021).

2.3.4 Dashboard (Tablero)

De ahora en adelante hablaré de los Dashboard por su nombre en español tableros. Los tableros permiten visualizar y ordenar los datos recibidos mostrándolos en widgets. (Espinosa & Orellana, 2021).

2.3.5 Widgets.

Son herramientas de visualización que permiten obtener el dato de una variable almacenada en un dispositivo y mostrarlo. Existen varios widgets lo que permite crear un ambiente llamativo para el usuario al momento de operar. (Espinosa & Orellana, 2021).

2.3.6 Eventos.

Ubidots permite enviar notificaciones de mensaje a quien lo necesite ycuando lo necesite, las alertas que activan los eventos se disparan cuando los valores de las variables a observar cruzan el valor antes preestablecido. (Espinosa & Orellana, 2021).

2.3.7 Creación de dispositivos

Los dispositivos son bloques para envío o recepción de datos a travésde un protocolo de conexión a la nube de Ubidots. Existen librerías que permiten la comunicación con diferentes plataformas como Arduino, Siemens, Microchip, Raspberry pi, etc, como se muestra en la figura 6.
En otras palabras, un dispositivo es una representación virtual de una fuente de datos. (Espinosa & Orellana, 2021).

La creación de un dispositivo dentro de Ubidots es automática. Ubidots se maneja por datos y la cuenta gratuita tiene un máximo de 4000 datos de envío y recepción. Luego de esa cantidad se debe esperar 24 horas para continuar con un intercambio de datos. (Espinosa & Orellana, 2021). En la siguiente figura se muestra la interfaz para la creación de dispositivos en el programa.

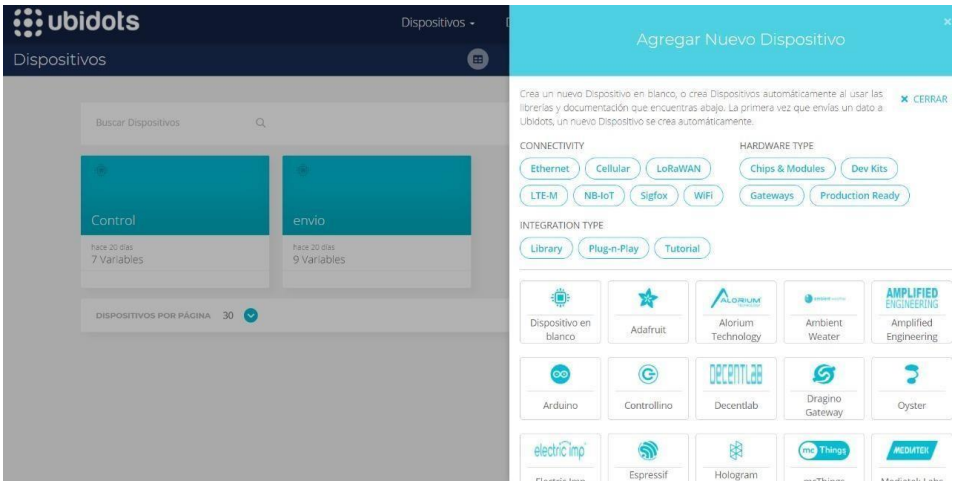


Figura 6: Interfaz creación de dispositivos en Ubidots. (Espinosa & Orellana, 2021).

2.3.8 Creación de Widgets

Una vez que se ha desarrollado un dispositivo, es crucial añadir widgets para interactuar con las variables del sistema, como se muestra en la figura 7. Al crear un widget, es necesario asignarle un nombre y elegir su tipo. Posteriormente, se pueden agregar otros datos opcionales, como la unidad de la variable y una descripción. Tal como se ilustra en la figura 8, se pueden configurar las propiedades del widget "Battery".

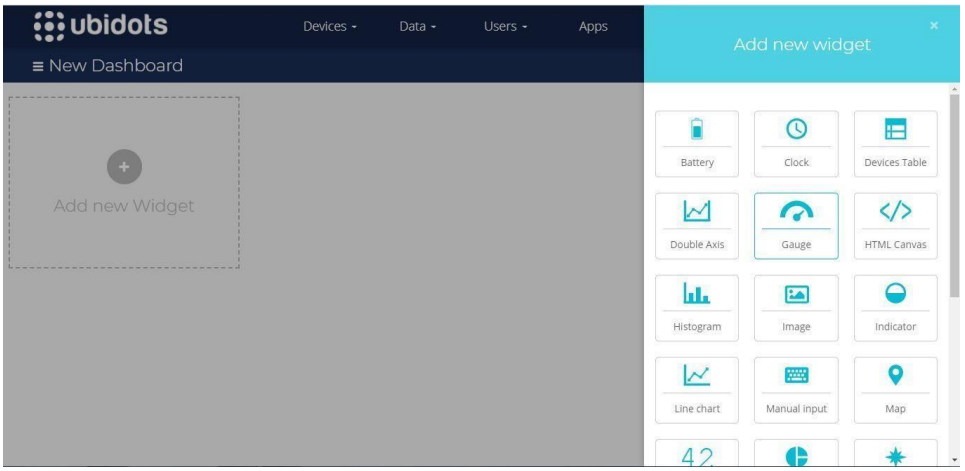


Figura 7: Interfaz de Ubidots (Espinosa & Orellana, 2021).

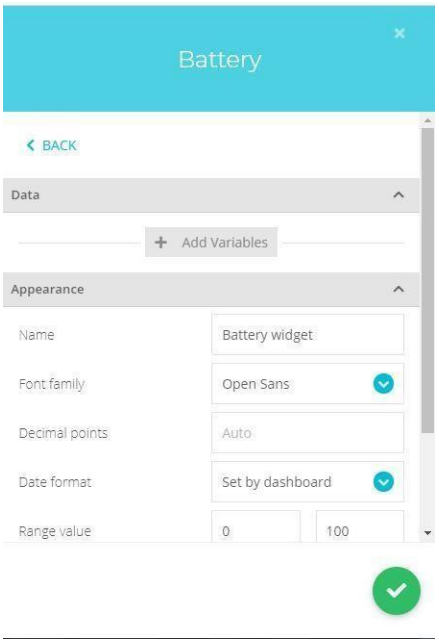


Figura 8: Creación de un variable dentro de Ubidots.

Ubidots permite agregar a cada tablero un máximo de widgets. Dentro del tablero se modifica su tamaño, así como permite ubicar la posiciónde los widgets (véase la figura 9). (Espinosa & Orellana, 2021).

Figura 9: Disposición variables tablero Ubidots
(Espinosa &Orellana, 2021).

2.3.9 Creación de variables

Las variables permiten administrar los datos dentro de Ubidots. Dentro de esta plataforma se pueden crear dos tipos de variables como lo había mencionado anteriormente: “Cruda”, “Sintética” como se observa en la figura 10 (Espinosa & Orellana, 2021).

Figura 10: Cuadro de diálogo para selección de tipos de variables dentro de Ubidots (Espinosa & Orellana, 2021).

Para acceder a una variable dentro de la nube, el nombre de la variable debe coincidir con “API Label” como se muestra en la figura 11, esta variable guardara todos los datos que se envíe o reciba durante su uso. (Espinosa & Orellana, 2021).

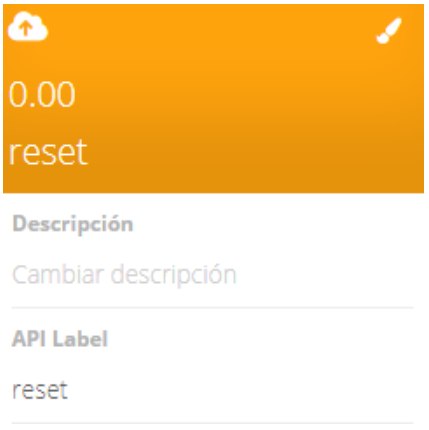


Figura 11: Envió y recepción de datos mediante una variable. (Espinosa & Orellana, 2021).

2.3.10 Compartir en Ubidots

Para compartir los widgets en forma de iframe, es necesario hacer clic en el botón "Compartir", el cual se encuentra ubicado en cada widget. Además, tienes la opción de compartir esta ventana con otros usuarios a través de un enlace. Ver figura 12. (Espinosa & Orellana, 2021).

Figura 12: Link para compartir iframe de la variable en Ubidots. (Espinosa &Orellana, 2021).

2.3.11 Token

Para poder enviar datos a una variable en Ubidots, se requiere acceso a la nube. Ubidots proporciona un token único para cada cuenta, el cual es necesario para realizar la conexión y el envío de datos. Este token se muestra en la figura 13. Este token caracteriza a la cuenta como única forma de encontrarla en la nube en otras palabras el token permite el acceso directo hacia la nube, el token se lo puede encontraren la sección de credenciales y tiene la opción de crear más de un token si es necesario. (Espinosa & Orellana, 2021).

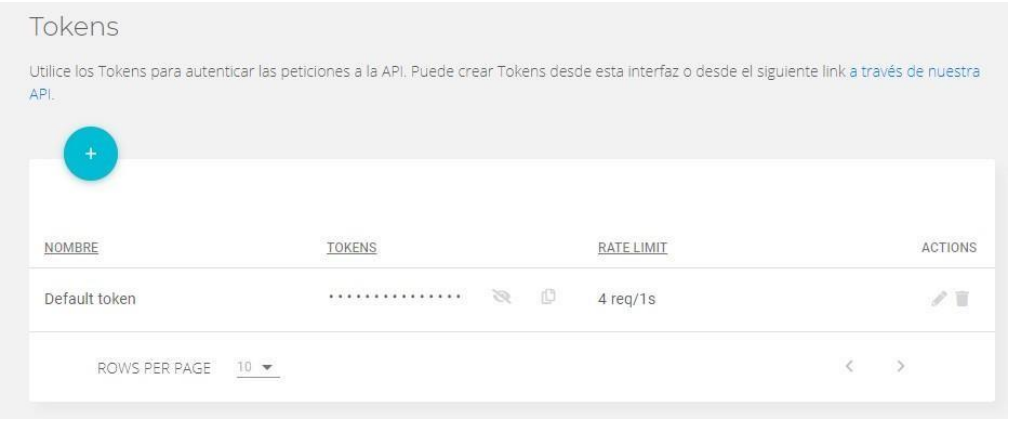


Figura 13: Obtención del Token. (Espinosa & Orellana, 2021).

El token está conformado por una serie de 36 números, letras ysímbolos especiales este es un ejemplo de un Token.

BBFF-Pi4613uPOxxxxx3BvPjgyUqSOiT6HD

2.4 Principales Dispositivos del Proyecto

Para el proyecto se utilizaron principalmente los siguientes dispositivos:

2.4.1 ESP32

El procesador ESP 32 pertenece a espressif Systems quienes son los creadores del ESP8266.

Figura 14*:* El microcontrolador ESP32
(Espinosa & Orellana,2021)

2.4.1.1 Especificaciones del ESP32

- Procesador dual core Xtensa LX6 de 32 bits.
- Es compatible Arduino con el plugin adecuado.
- Velocidad de reloj: Entre 160 Mhz y 240 Mhz.
- 520 Kb de RAM.
- Wifi integrado: Acces point & Stantion.
- Bluetooth 4.2 2.4 Ghz; BT 2.0 Y 4.0 BLE.
- 36 GPIO pins.
- 16 x Analog-to-Digital Converter (ADC) de 12 bits de resolución y se pueden programar con límite de entrada a 1V, 2V y 4V.
- 2 x Digital to Analog converter DAC de 8 bits.
- Pueden definirse hasta 16 canales de PWM.
- 2 x UART o puertas serie.
- 2 x I2C channels y 4 x SPI channels.

2.4.1.2 Terminales del ESP32 (Pinout)

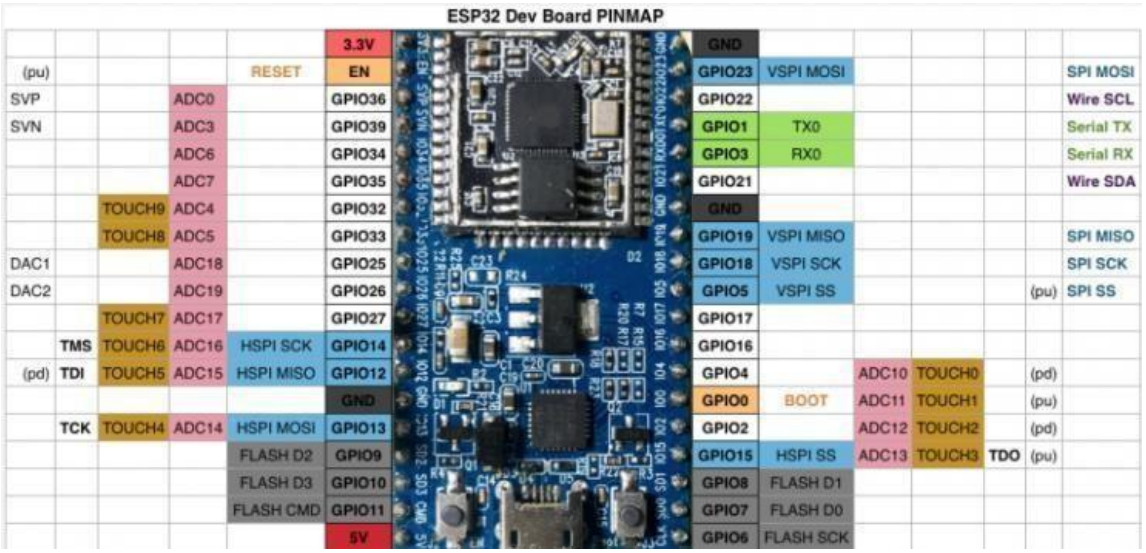


Figura 15: Terminales del ESP32 (Espinosa & Orellana, 2021).

2.4.2 Sensor de Temperatura y Humedad DHT11

El DHT11 es un sensor que mide temperatura y humedad relativa. Se caracteriza por tener la señal digital calibrada por lo que asegura una alta fiabilidad y estabilidad a largo plazo. Además, contiene un microcontrolador de 8 bits integrado ofreciendo excelente calidad, respuesta rápida y gran relación precio-efectividad. (Iraceburu & Goicoechea, 2014).

Figura 16: Sensor de Temperatura y Humedad DHT11
(Espinosa & Orellana,2021).

El dispositivo está compuesto por dos sensores resistivos: uno de humedad y otro NTC (Coeficiente de Temperatura Negativa). Es capaz de medir la humedad relativa en un rango que va desde el 20% hasta el 95%, así como la temperatura en un intervalo de 0ºC a 50ºC. Utiliza un protocolo de comunicación a través de un único hilo (1-wire), lo que facilita su integración en proyectos de manera rápida y sencilla. Además, es compacto, tiene un bajo consumo de energía y su señal puede alcanzar hasta 20 metros de distancia. Su principal problema es quetan sólo proporciona medidas enteras, pues tiene una resolución de 1ºC para las temperaturas y del 1% para la humedad relativa. (Iraceburu & Goicoechea, 2014).

2.4.3 Driver L298N

El módulo controlador de motores **L298N** H-bridge permite controlar la velocidad

y la dirección de dos motores de corriente continua o un motor paso a paso de una forma muy sencilla, gracias a los 2 los dos H-bridge que monta. (Protelec, 2023).

Ya se ha hablado de ellos antes, pero básicamente un puente-H o H-bridge es un componente formado por 4 transistores que permite invertir el sentido de la corriente, y de esta forma se puede invertir el sentido de giro del motor. (Protelec, 2023).

El rango de tensiones en el que trabaja este módulo va desde 3V hasta 35V, y una intensidad de hasta 2A. A la hora de alimentarlo hay que tener en cuenta que la electrónica del módulo consume unos 3V, así que los motores reciben 3V menos que la tensión con la que alimenta el módulo. (Protelec, 2023).

El L298N incluye un regulador de tensión que permite obtener del módulo una tensión de 5V, perfecta para alimentar el Arduino, como se observar en la figura 17 las partes que tiene un Driver L298N. Eso sí, este regulador sólo funciona si alimenta el módulo con una tensión máxima de 12V. Es un módulo que se utiliza mucho en proyectos de robótica, por su facilidad de uso ysu reducido precio. (Protelec, 2021).

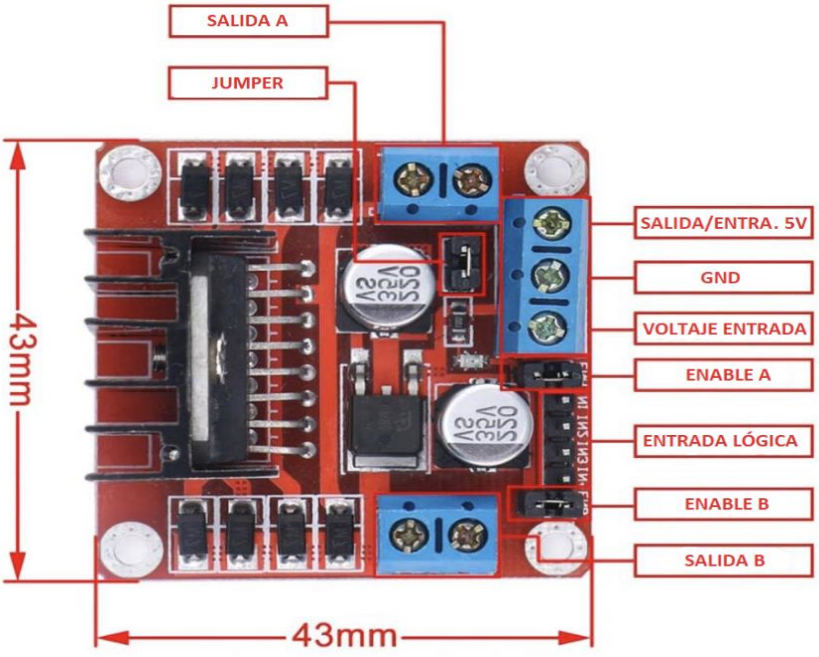


Figura 17: Driver L298N (Joober, 2021).

3. MARCO METODOLOGICO

3.1 Circuito general del proyecto

En la figura 18 se muestra el circuito general del controlador de temperatura para el galpón de pollos.

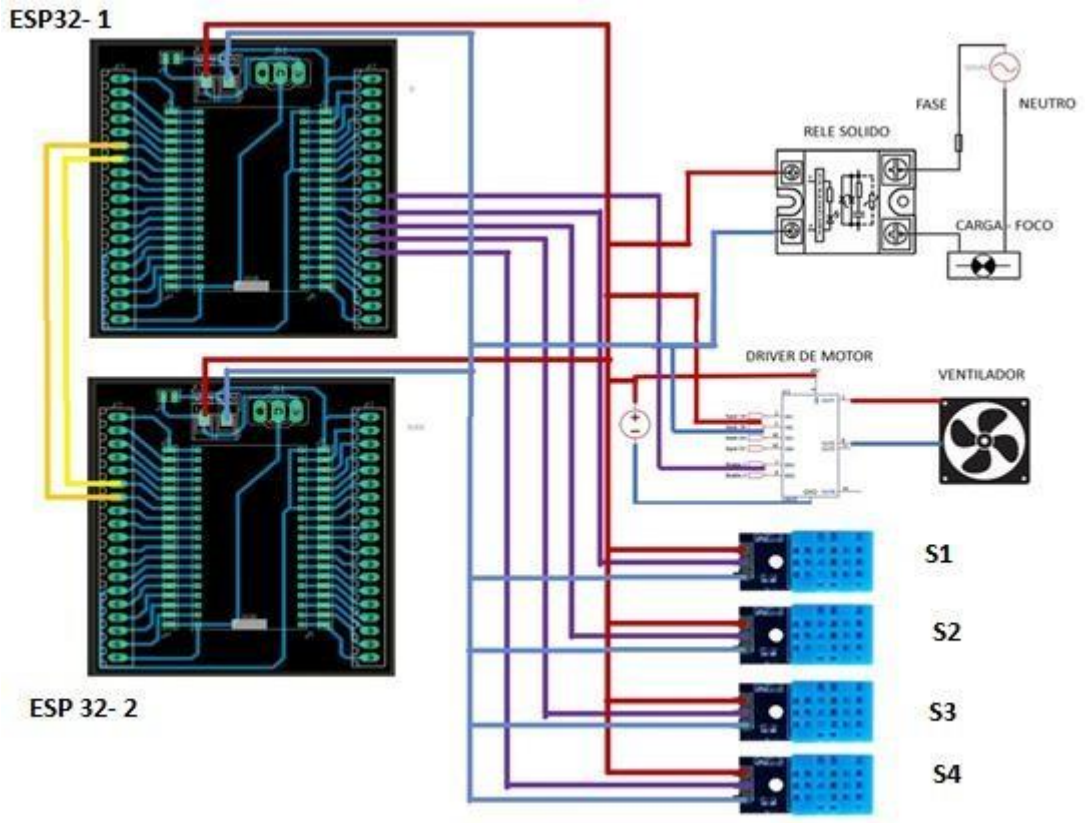


Figura 18: Circuito general del proyecto.

Se muestra el circuito para el control de temperatura del galpón para un criadero de pollos la cual está constituida por dos microcontroladores ESP32 .En el primer microcontrolador (ESP32 – 1) , se procesan las señales detemperatura de los sensores S1 , S2 , S3 y S4 que según el correspondiente programa en IDE el cual lleva incluido la red Adaline , va a emitir la salida para el relay de estado sólido que acciona al foco led de potencia para mantener unatemperatura según el valor de referencia o setpoint.

Un ventilador va a operar desde el momento en que se encienda el sistema con la ayuda del driver L298N.

Tanto la señal de salida para el foco como para el ventilador provienen del microcontrolador ESP32 – 1.

El microcontrolador ESP32 – 2, procesa el internet de las cosas (IoT) Ubidots para que se pueda visualizar y monitorear las variaciones de temperatura, y accionamiento del foco.

3.2 Maqueta del Proyecto

En la figura 19 se muestra la maqueta del galpón con control de temperatura para el criadero de pollos.
Las partes son las siguientes:

1	Cabina
2,3,4, y 5	Sensores de temperatura y humedad DHT11
6	Ventilador de 12 V
7	Led de potencia 12 V
8	Fuente de voltaje DC
9	Driver L298N para el ventilador
10	Relé de estado sólido
11 y 12	Microcontrolador ESP32

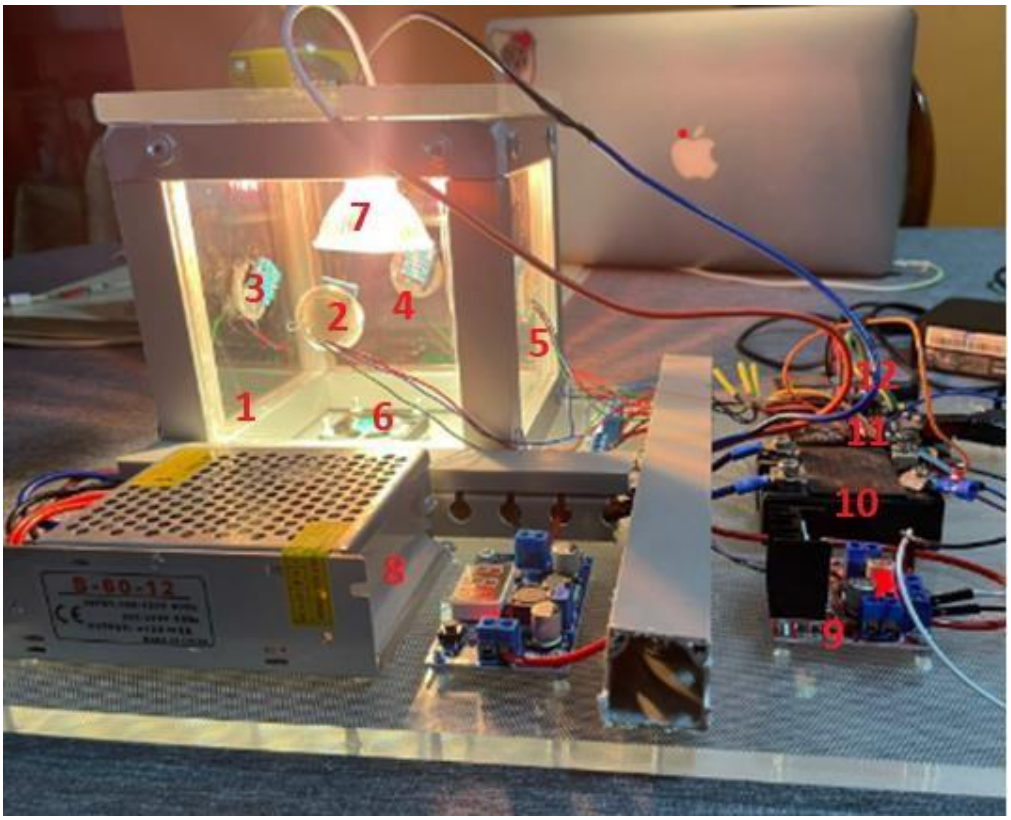


Figura 19: Maqueta del Proyecto.

En la figura 20 se observan los dos microcontroladores ESP32.

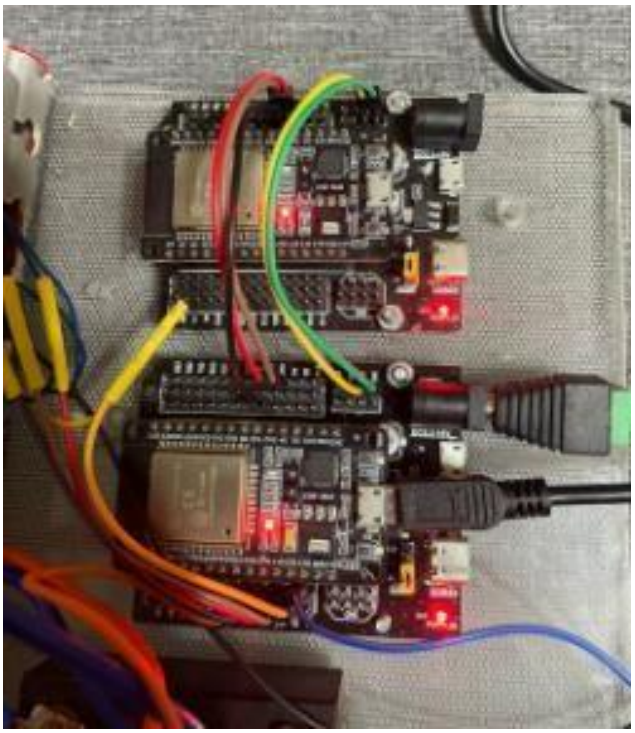


Figura 20: Microprocesadores ESP32.

En la figura 21 se aprecia el Driver L298N para el accionamiento del ventilador

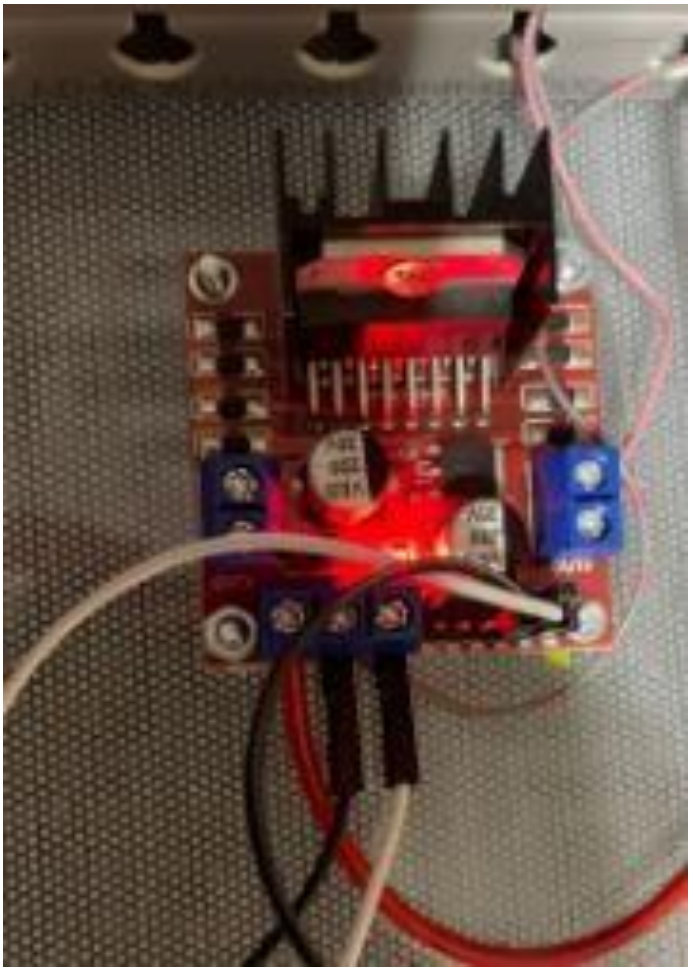
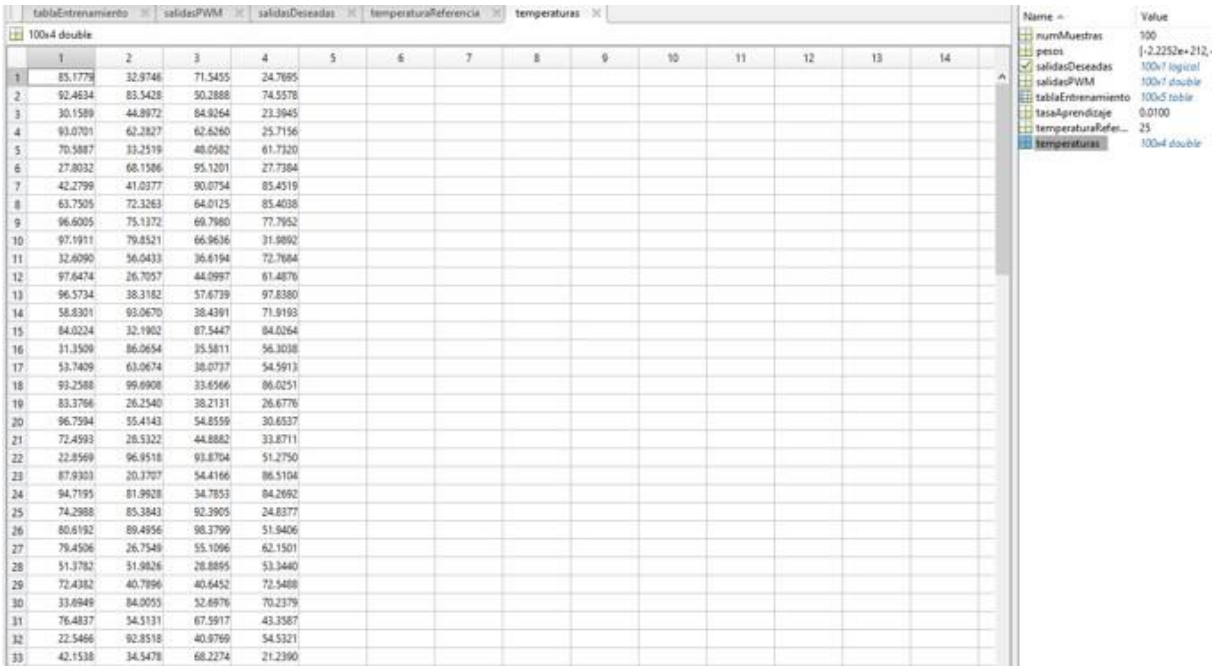


Figura 21: Driver L298N.

3.3 Programa del Control de Temperatura

A continuación, se muestra los programas que se realizaron para la elaboración del sistema de control de temperatura para el criadero de pollos, los cuales se componen de un entrenamiento de las temperaturas utilizando el software Matlab, configuración del sistema utilizando el Ide de Arduino y el monitoreo con el IoT Ubidots.

Procesamiento de datos para el entrenamiento de temperaturas utilizando Matlab.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	85.1779	32.9746	71.5455	24.7685										
2	92.4634	83.5428	50.2888	74.5578										
3	30.1589	44.8972	84.9264	23.3945										
4	93.0701	62.2827	62.6260	25.7156										
5	70.5887	33.2519	48.0582	61.7320										
6	27.8032	68.1586	95.1201	27.7384										
7	42.2799	41.0377	90.0754	85.4519										
8	63.7505	72.3263	64.0125	85.4038										
9	96.6005	75.1372	69.7980	77.7952										
10	97.1911	79.8521	66.9636	31.9892										
11	32.6090	56.0433	36.6194	72.7684										
12	97.6474	26.7057	44.0997	61.4876										
13	96.5734	38.3182	57.6739	97.8380										
14	58.8301	93.0670	38.4391	71.9193										
15	84.0224	32.1902	87.5447	84.0264										
16	31.3509	86.0654	35.5811	56.3038										
17	53.7409	63.0674	38.0737	54.5913										
18	93.2588	99.6908	33.6566	86.0251										
19	83.3766	26.2540	38.2131	26.6776										
20	96.7594	55.4143	54.8559	30.6537										
21	72.4593	28.5322	44.8882	33.8711										
22	22.8569	96.9518	93.8704	51.2750										
23	87.9303	20.3707	54.4166	86.5104										
24	94.7195	81.9928	34.7853	84.2692										
25	74.2988	85.3843	92.3905	24.8377										
26	80.6192	89.4956	98.3799	51.9406										
27	79.4506	26.7549	55.1096	62.1501										
28	51.3782	51.9826	28.8895	53.3440										
29	72.4382	40.7896	40.6452	72.5488										
30	33.6949	84.0055	52.6976	70.2379										
31	76.4837	54.5131	67.5917	43.3587										
32	22.5486	92.8518	40.9769	54.5321										
33	42.1538	34.5478	68.2274	21.2390										

Figura 22: Datos de temperatura en Matlab.

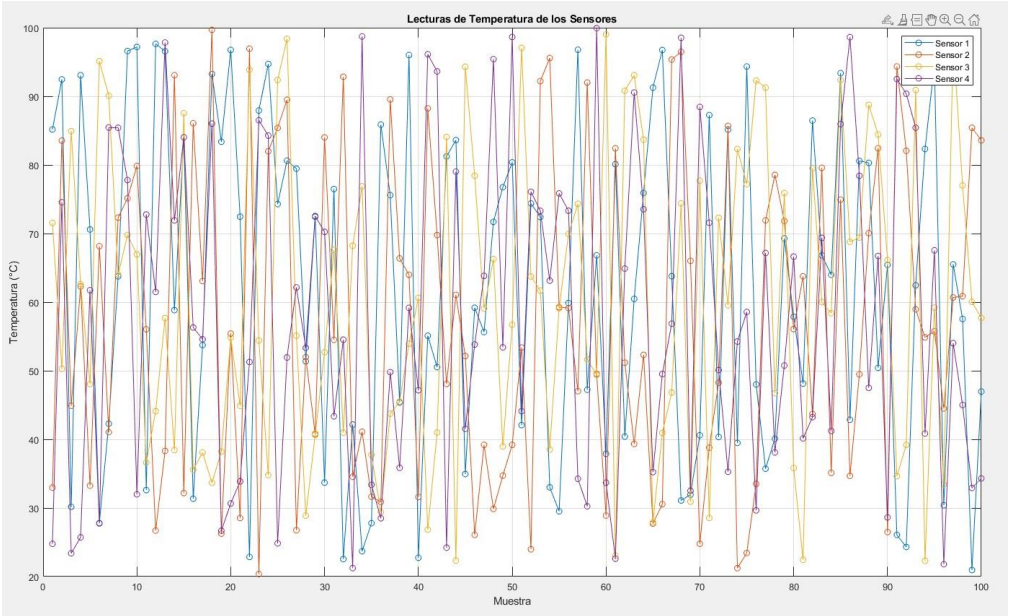


Figura 23: Lecturas de temperatura de los sensores.

En la figura 23 se observa las lecturas de los datos mediante un control en lazo abierto hasta que se estabilicen. En Matlab se crea una tabla llamada Tabla de

entrenamiento con 100 datos y la salida de pwm enviada. (Fig. 24)

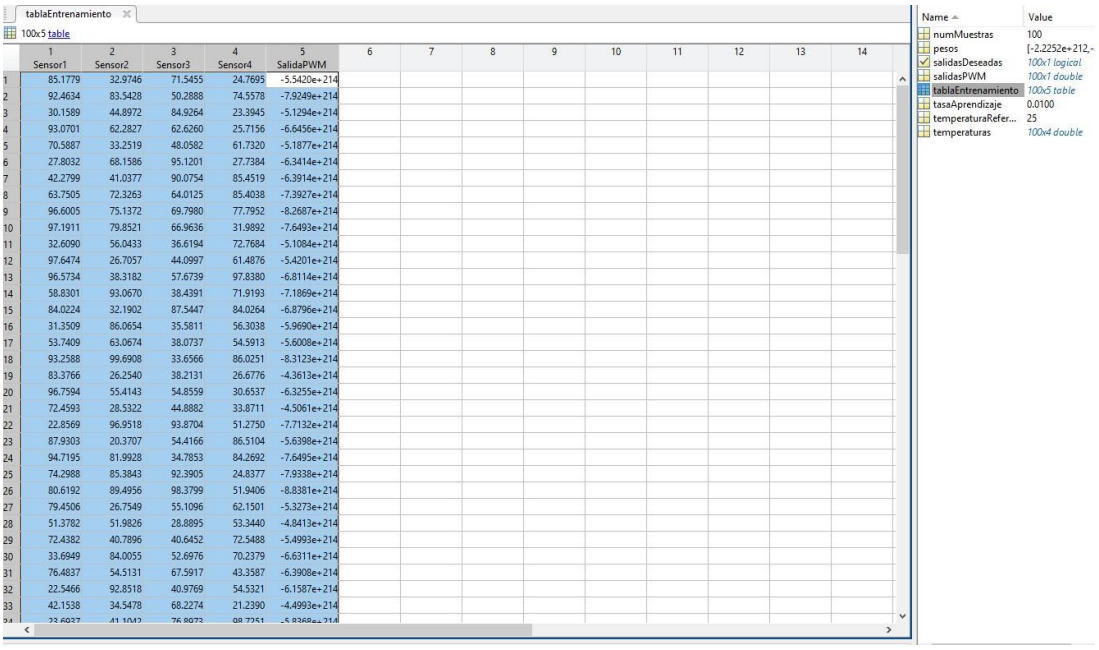


Figura 24: Tabla de entrenamiento.

```
% Parámetros de simulación
numMuestras = 100;
temperaturaReferencia = 25; % Asegúrate de que esté en el rango de 20 a 100
tasaAprendizaje = 0.01;

% Generar datos
[temperaturas, salidasDeseadas] = Entrenamiento(numMuestras, temperaturaReferencia);

% Entrenar la red Adaline
pesos = entrenarAdaline(temperaturas, salidasDeseadas, tasaAprendizaje);
```

Figura 25: Entrenamiento red Adaline.

En la figura 25 se muestra la instrucción para definir los parámetros de simulación, generación de datos y entrenamiento de la red Adaline.

```
function [temperaturas, salidasDeseadas] = Entrenamiento(numMuestras, temperaturaReferencia)
    % Rango de temperaturas
    minTemperatura = 20;
    maxTemperatura = 100;
    temperaturas = minTemperatura + (maxTemperatura - minTemperatura) * rand(numMuestras, 4);
    salidasDeseadas = mean(temperaturas, 2) < temperaturaReferencia;
end
```

Figura 26: Rangos máximos y mínimos.

En la figura 26 se muestran el promedio de las muestras con su rango máximo y mínimo establecidos por el usuario.

```
function pesos = entrenarAdaline(temperaturas, salidasDeseadas, tasaAprendizaje)
    pesos = randn(1, 4);
    numMuestras = size(temperaturas, 1);
    for i = 1:numMuestras
        entrada = temperaturas(i, :);
        salidaDeseada = salidasDeseadas(i);
        salidaPWM = pesos * entrada';
        error = salidaDeseada - salidaPWM;
        pesos = pesos + tasaAprendizaje * error * entrada;
    end
end
```

Figura 27: Tasa de aprendizaje.

En el entrenamiento se procede a realizar un lazo cerrado hasta llegar al valor total de muestra y que su valor de entrenamiento sea la meta establecida como tasa de aprendizaje. (Fig. 27).

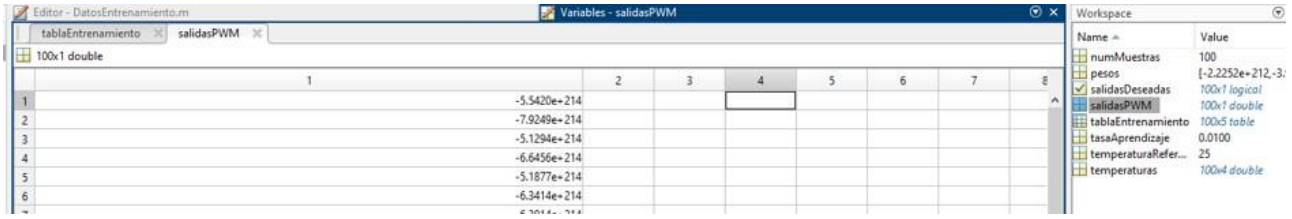


Figura 28: Salida PWM.

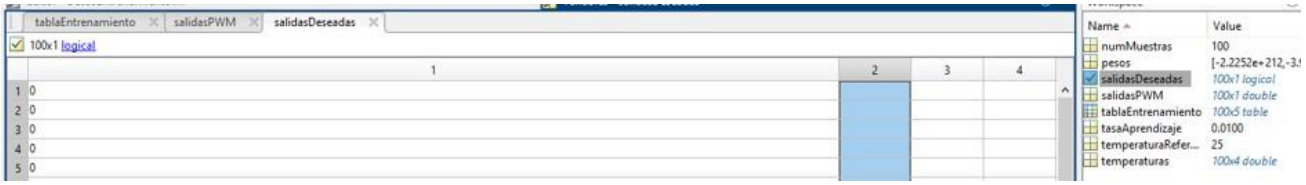


Figura 29: Salida Deseada.

Comparación entre salidas PWM del entrenamiento (Fig. 28) vs a los datos de entrenamiento con la PWM deseada (Fig. 29) cuando su error sea cero.

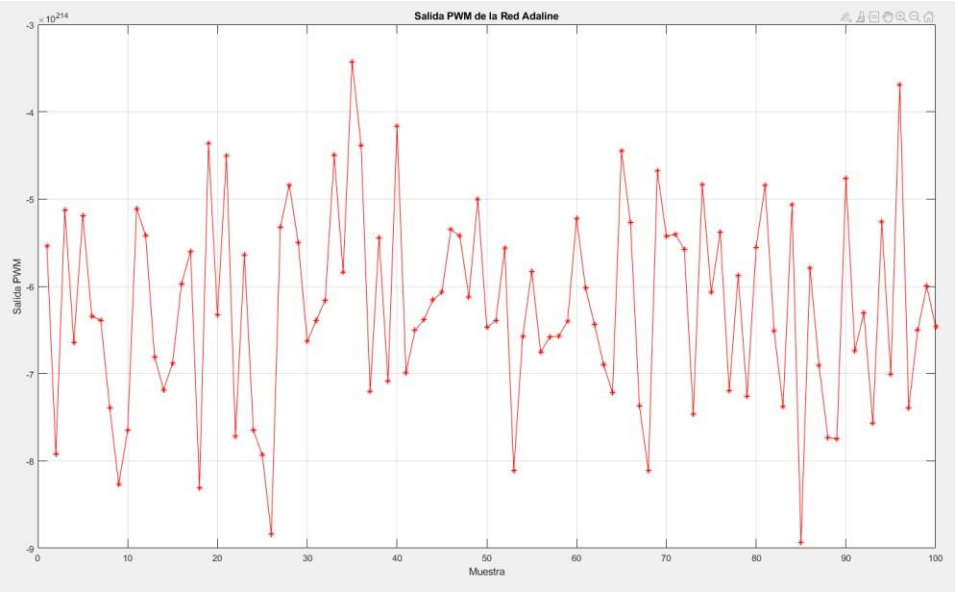


Figura 30: Salida PWM Red Adaline.

Figura 31: Rango de Pesos.

Posterior al entrenamiento de las temperaturas, se obtienen los rangos de pesos. (Fig. 30).

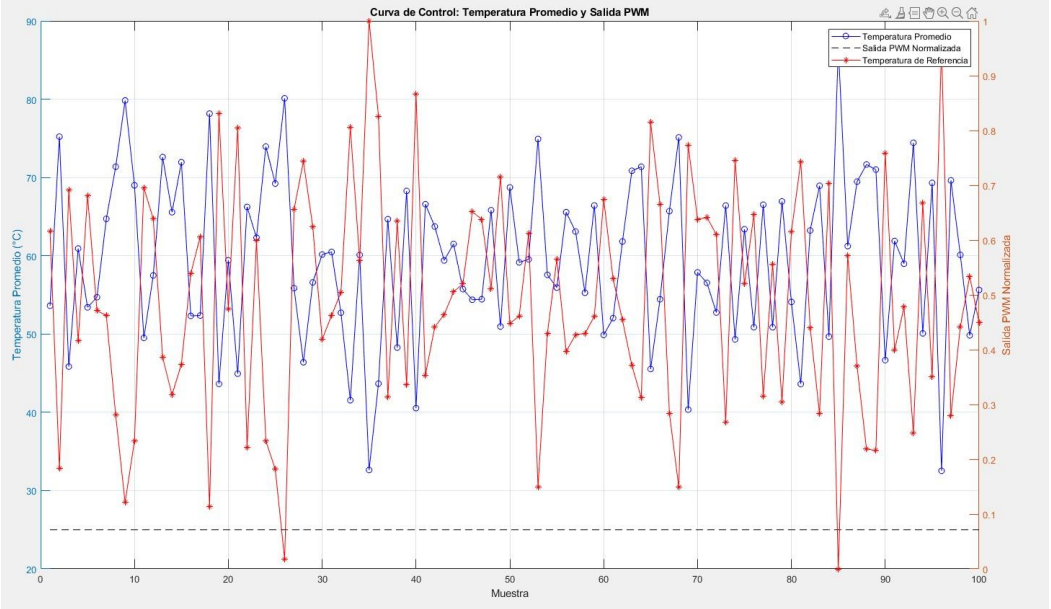


Figura 32: Grafico de control temperatura promedio y salida PWM.

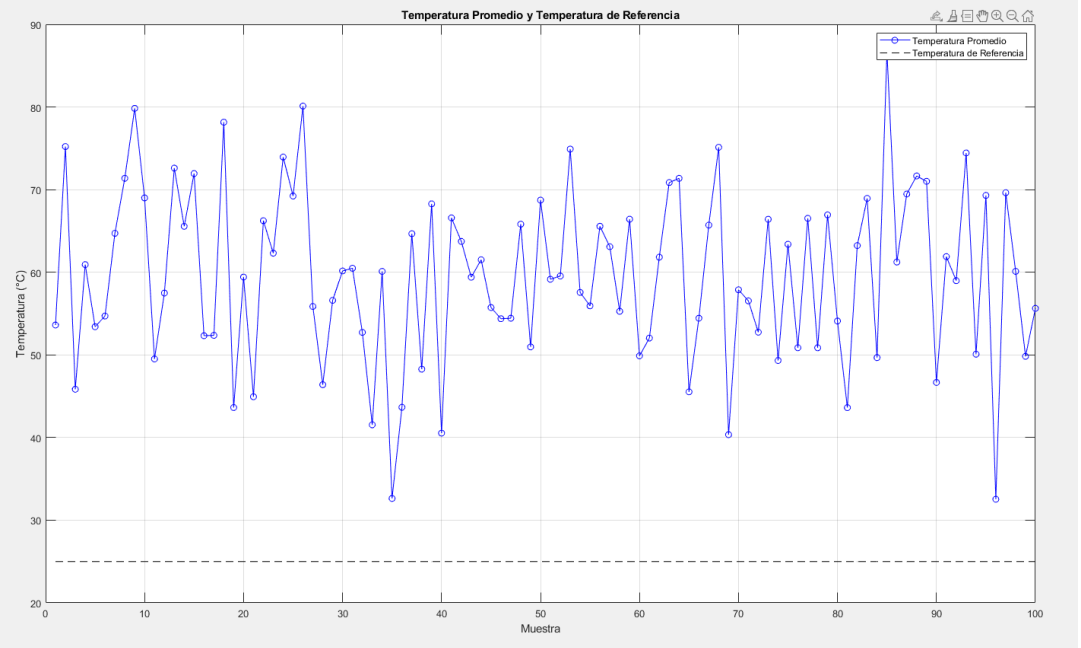


Figura 33: Grafico de control temp. promedio y temp. de referencia.

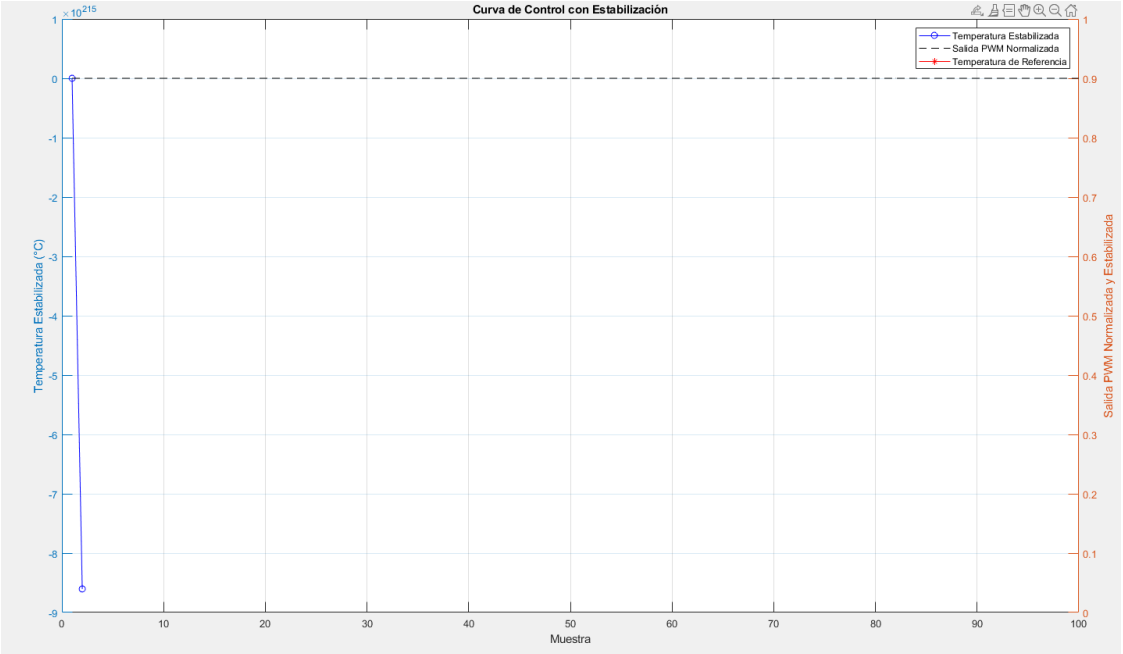


Figura 34: Grafico de control de estabilización del sistema.

A continuación, se describe el programa para el control de la red Adaline, especificando sus entradas y salidas:

```
// Inclusión de bibliotecas necesarias
#include <Adafruit_Sensor.h> // Biblioteca genérica de Adafruit para sensores
#include "DHT.h"             // Biblioteca para el manejo de sensores DHT

#include <HardwareSerial.h>  // Biblioteca para comunicación serial
HardwareSerial mySerial(2);  // Inicialización de comunicación serial en el puerto 2

// Definición de constantes y variables globales
const int pinSalidaPWM = 25;  // Pin para la salida PWM
#define DHTPIN_1 12           // Pin del sensor DHT 1
#define DHTPIN_2 14           // Pin del sensor DHT 2
#define DHTPIN_3 27           // Pin del sensor DHT 3
#define DHTPIN_4 26           // Pin del sensor DHT 4
#define DHTTYPE DHT11         // Define el tipo de sensor DHT como DHT11

// Inicialización de los sensores DHT
DHT dht1(DHTPIN_1, DHTTYPE);
DHT dht2(DHTPIN_2, DHTTYPE);
DHT dht3(DHTPIN_3, DHTTYPE);
DHT dht4(DHTPIN_4, DHTTYPE);

// Variables para el modelo Adaline
float peso;  // Peso de la red Adaline
float tasaAprendizaje = 0.01;  // Tasa de aprendizaje para el ajuste de peso
float temperaturaReferencia = 30.0; // Temperatura de referencia

// Variables para almacenar datos de los sensores
float h1, t1, f1, hif1, hic1; // Humedad y temperatura del sensor 1
float h2, t2, f2, hif2, hic2; // Humedad y temperatura del sensor 2
float h3, t3, f3, hif3, hic3; // Humedad y temperatura del sensor 3
float h4, t4, f4, hif4, hic4; // Humedad y temperatura del sensor 4
```

Se definen los cuatro sensores DHT11, también se especifica la salida de modulación de ancho de pulso PWM.

A continuación, se define la selección del valor de referencia de temperatura la cual se realiza con la ayuda de un potenciómetro:

```
const int potPin = 34;

// variable for storing the potentiometer value
int potValue = 0;
void setup() {
  Serial.begin(115200);        // Inicia la comunicación serial con la computadora
  mySerial.begin(9600);        // Inicia la comunicación serial en el puerto 2
  dht1.begin();                // Inicia el sensor DHT 1
  dht2.begin();                // Inicia el sensor DHT 2
  dht3.begin();                // Inicia el sensor DHT 3
  dht4.begin();                // Inicia el sensor DHT 4
  peso = random(1000) / 1000.0 - 0.5;    // Inicializa el peso de manera aleatoria


  pinMode(pinSalidaPWM, OUTPUT); // Configura el pin de salida PWM como salida
  digitalWrite(pinSalidaPWM, LOW); // Establece el pin de salida PWM a bajo

}

   // Función para leer la temperatura de un sensor DHT
   float leerTemperatura(DHT& sensor) {
     float temp = sensor.readTemperature();
     if (isnan(temp)) {
       Serial.println("Error leyendo la temperatura del sensor");
       return 0.0;
     }
     return temp;
   }

   // Función para calcular la temperatura promedio de todos los sensores DHT
   float leerTemperaturaPromedio() {
     float sumaTemperaturas = leerTemperatura(dht1) + leerTemperatura(dht2) +
                              leerTemperatura(dht3) + leerTemperatura(dht4);
     return sumaTemperaturas / 4;
   }
void loop() {
  int offset = 1;
  int bias = -0.1;  int valorPWM ;
 potValue = analogRead(potPin);
 temperaturaReferencia = map(potValue, 0, 4095, 30, 45);



 //    minTemperatura = 20;
 //    maxTemperatura = 100;
 //    temperaturas = minTemperatura + (maxTemperatura - minTemperatura) * rand(numMuestras, 4);
 //    salidasDeseadas = mean(temperaturas, 2) < temperaturaReferencia;


 float temperaturaPromedio = leerTemperaturaPromedio() ; // Calcula la temperatura promedio
 float error = temperaturaReferencia - temperaturaPromedio; // Calcula el error
//  con respecto a la temperatura de referencia
```

```
// Calcula la salida de la red Adaline
float salidaAdaline = error * peso;

// Ajusta el peso utilizando el algoritmo de aprendizaje Adaline
//  pesos = randn(1, 4);
//  numMuestras = size(temperaturas, 1);
//for i = 1 : numMuestras
//          entrada = temperaturas(i, : );
//  salidaDeseada = salidasDeseadas(i);
//  salidaPWM = pesos * entrada';
//              error = salidaDeseada - salidaPWM;
//              pesos = pesos + tasaAprendizaje * error * entrada;

peso += tasaAprendizaje * error * temperaturaPromedio + bias;

peso = constrain(peso, -1, 1); // Restringe el peso entre -1 y 1
// Convierte la salida de Adaline a un valor PWM y lo ajusta en el pin correspondiente
if (error < 0) {
  valorPWM = constrain(salidaAdaline, -1, 1);
  valorPWM = map(valorPWM, -1, 1, 255, 0);
  analogWrite(pinSalidaPWM, valorPWM);
}
else {
  valorPWM = constrain(salidaAdaline, -1, 1);
  valorPWM = map(valorPWM, -1, 1, 0, 255);
  analogWrite(pinSalidaPWM, valorPWM);
}

// Lee los datos de humedad y temperatura de cada sensor DHT
h1 = dht1.readHumidity(); t1 = dht1.readTemperature(); f1 = dht1.readTemperature(true);
h2 = dht2.readHumidity(); t2 = dht2.readTemperature(); f2 = dht2.readTemperature(true);
h3 = dht3.readHumidity(); t3 = dht3.readTemperature(); f3 = dht3.readTemperature(true);
h4 = dht4.readHumidity(); t4 = dht4.readTemperature(); f4 = dht4.readTemperature(true);

// Espera no bloqueante
unsigned long tiempoEspera = 500; // 1000 milisegundos
static unsigned long ultimoTiempo = 0;
unsigned long tiempoActual = millis();
if (tiempoActual - ultimoTiempo >= tiempoEspera) {
  ultimoTiempo = tiempoActual;
```

```
// Envio al serial para publicar
mySerial.print("S1:");  mySerial.print(h1); mySerial.print(",");
mySerial.print("S2:");  mySerial.print(h2); mySerial.print(",");
mySerial.print("S3:");  mySerial.print(h3); mySerial.print(",");
mySerial.print("S4:");  mySerial.print(h4); mySerial.print(",");
mySerial.print("S5:");  mySerial.print(t1 - offset); mySerial.print(",");
mySerial.print("S6:");  mySerial.print(t2 - offset); mySerial.print(",");
mySerial.print("S7:");  mySerial.print(t3 - offset); mySerial.print(",");
mySerial.print("S8:");  mySerial.print(t4 - offset); mySerial.print(",");
mySerial.print("S9:");  mySerial.print(temperaturaPromedio - offset); mySerial.print(",");
mySerial.print("S10:");  mySerial.print(error); mySerial.print(",");
mySerial.print("S11:");  mySerial.print(salidaAdaline); mySerial.print(",");
mySerial.print("S12:");  mySerial.println(peso);

//
//    Serial.print("S1:");  Serial.print(h1); Serial.print(",");
//    Serial.print("S2:");  Serial.print(h2); Serial.print(",");
//    Serial.print("S3:");  Serial.print(h3); Serial.print(",");
//    Serial.print("S4:");  Serial.print(h4); Serial.print(",");
Serial.print("REFERENCIA:");  Serial.print(temperaturaReferencia); Serial.print(",");
Serial.print("T1:");  Serial.print(t1 - offset ); Serial.print(",");
Serial.print("T2:");  Serial.print(t2 - offset); Serial.print(",");
Serial.print("T3:");  Serial.print(t3 - offset ); Serial.print(",");
Serial.print("T4:");  Serial.print(t4 - offset ); Serial.print(",");
Serial.print("PROMEDIO:");  Serial.print(temperaturaPromedio - offset ); Serial.print(",");
Serial.print("ERROR:");  Serial.print(error); Serial.print(",");
Serial.print("SALIDA_ADALINE:");  Serial.print(salidaAdaline); Serial.print(",");
Serial.print("PESO:");  Serial.println(peso);
```

3.4 Programa para la utilización del IoT Ubidots

```
/****************************************
   Include Libraries
 ****************************************/
#include "UbidotsEsp32Mqtt.h" // Incluye la librería para conectar con Ubidots mediante MQTT.
#include <HardwareSerial.h>   // Incluye la librería para comunicación serial.
HardwareSerial mySerial(2);  // Crea un objeto Serial en el puerto 2.

/****************************************
   Define Constants
 ****************************************/
// Constantes para la conexión con Ubidots y la red Wi-Fi.
const char *UBIDOTS_TOKEN = "BBUS-NWExm09CJoF0bIbczvY4qXROsMz9SI";           // Token de Ubidots
const char *WIFI_SSID = "invernadero";                // SSID del Wi-Fi.
const char *WIFI_PASS = "criadero";               // Contraseña del Wi-Fi.
// Etiquetas para los dispositivos y variables en Ubidots.
const char *PUBLISH_DEVICE_LABEL = "CRIADERO"; // Put here your Device label to which data  will be published
const char *PUBLISH_VARIABLE_LABEL_1 = "H_1";
const char *PUBLISH_VARIABLE_LABEL_2 = "H_2";
const char *PUBLISH_VARIABLE_LABEL_3 = "H_3";
const char *PUBLISH_VARIABLE_LABEL_4 = "H_4";
const char *PUBLISH_VARIABLE_LABEL_5 = "T_1";
const char *PUBLISH_VARIABLE_LABEL_6 = "T_2";
const char *PUBLISH_VARIABLE_LABEL_7 = "T_3";
const char *PUBLISH_VARIABLE_LABEL_8 = "T_4";
const char *PUBLISH_VARIABLE_LABEL_9 = "PROMEDIO";
const char *PUBLISH_VARIABLE_LABEL_10 = "ERROR";
const char *PUBLISH_VARIABLE_LABEL_11 = "ADALINE";
const char *PUBLISH_VARIABLE_LABEL_12 = "PESO";
```

En esta sección se define el Token. Se define la red Wifi y se coloca el password.

```
// Frecuencia de publicación y configuración de sensores.
const int PUBLISH_FREQUENCY = 5000; // Frecuencia de publicación en milisegundos.
const int num_sensores = 12;        // Número de sensores.
float sensores[12];                 // Arreglo para almacenar los datos de los sensores.
unsigned long timer;                // Temporizador para controlar la publicación.

// Variables para sensores de pH.
float ph1, ph2, ph3, ph4; // Variables para almacenar los valores de pH.


Ubidots ubidots(UBIDOTS_TOKEN); // Crea un objeto Ubidots con el token proporcionado.


/****************************************
   Auxiliar Functions
 ****************************************/
// Función de retrollamada para mensajes MQTT.
void callback(char *topic, byte *payload, unsigned int length)
{
  String mensaje = "Message arrived ["; // Construye un string con el mensaje recibido.
  mensaje += topic; // Añade el tema del mensaje.
  mensaje += "] ";
  for (int i = 0; i < length; i++) {
    mensaje += (char)payload[i]; // Añade el payload del mensaje.
  }
  mensaje += "\n";
}
/****************************************

/****************************************
   Main Functions
 ****************************************/

void setup()
{
  Serial.begin(115200);             // Inicia la comunicación serial con la computadora.
  mySerial.begin(9600);             // Inicia la comunicación serial con otros dispositivos.
  ubidots.connectToWifi(WIFI_SSID, WIFI_PASS); // Conecta el dispositivo a Wi-Fi.
  ubidots.setCallback(callback);    // Establece la función de retrollamada para MQTT.
  ubidots.setup();                  // Configura el objeto Ubidots.
  ubidots.reconnect();              // Reconecta con Ubidots si es necesario.
  timer = millis();                 // Inicializa el temporizador.
}

// Función loop, se ejecuta repetidamente.
void loop()
{
  datos();    // Llama a la función que lee los datos de los sensores.
  principal(); // Llama a la función que maneja la conexión con Ubidots y la publicación de datos.
}

// Función para leer datos de los sensores.
void datos()
{
  if (mySerial.available() > 0) { // Verifica si hay datos disponibles en el puerto serial.
    String datos = mySerial.readStringUntil('\n'); // Lee los datos como un string hasta el salto de línea.
```

```
    // Extrae los valores de los sensores del string.
    for (int i = 0; i < num_sensores; i++) {
      String identificador = "S" + String(i + 1) + ":"; // Identificador del sensor.
      int inicio_valor = datos.indexOf(identificador) + identificador.length(); // Encuentra el inicio del valor.
      int fin_valor = datos.indexOf(',', inicio_valor); // Encuentra el final del valor.
      float valor = datos.substring(inicio_valor, fin_valor).toFloat(); // Convierte el valor a float y lo almacena.
      sensores[i] = valor; // Almacena el valor en el arreglo.
    }
  }
}

void principal()
{
  if (!ubidots.connected()) { // Verifica si el dispositivo está conectado a Ubidots.
    ubidots.reconnect();      // Intenta reconectar si no está conectado.
  }
  if (abs(static_cast<long>(millis() - timer)) > PUBLISH_FREQUENCY) // triggers the routine every 5 seconds
  {

    ubidots.add(PUBLISH_VARIABLE_LABEL_1, sensores[0]);
    ubidots.add(PUBLISH_VARIABLE_LABEL_2, sensores[1]);
    ubidots.add(PUBLISH_VARIABLE_LABEL_3, sensores[2]);
    ubidots.add(PUBLISH_VARIABLE_LABEL_4, sensores[3]);

     ubidots.add(PUBLISH_VARIABLE_LABEL_5, sensores[4]);
     ubidots.add(PUBLISH_VARIABLE_LABEL_6, sensores[5]);
     ubidots.add(PUBLISH_VARIABLE_LABEL_7, sensores[6]);
     ubidots.add(PUBLISH_VARIABLE_LABEL_8, sensores[7]);
     ubidots.publish(PUBLISH_DEVICE_LABEL);
     ubidots.add(PUBLISH_VARIABLE_LABEL_9, sensores[8]);
     ubidots.add(PUBLISH_VARIABLE_LABEL_10, sensores[9]);
     ubidots.add(PUBLISH_VARIABLE_LABEL_11, sensores[10]);
     ubidots.add(PUBLISH_VARIABLE_LABEL_12, sensores[11]);
     ubidots.publish(PUBLISH_DEVICE_LABEL);
     timer = millis();
   }
   ubidots.loop();
}
String obtenerEtiqueta(String mensaje) {
  int inicioEtiqueta = mensaje.indexOf("/B_") + 4;
  int finEtiqueta = mensaje.indexOf("/lv]");

  if (inicioEtiqueta >= 0 && finEtiqueta >= 0 && finEtiqueta > inicioEtiqueta) {
    return mensaje.substring(inicioEtiqueta, finEtiqueta);
  } else {
    return "";
  }
}
String obtenerValor(String mensaje) {
  int inicioValor = mensaje.lastIndexOf("]") + 2;

  if (inicioValor >= 0 && inicioValor < mensaje.length()) {
    return mensaje.substring(inicioValor);
  } else {
    return "";
  }
```

4. RESULTADOS

En esta sección se van a mostrar las pruebas realizadas en el proyecto de control de temperatura de un galpón para criadero de pollos utilizando la Red Adaline.

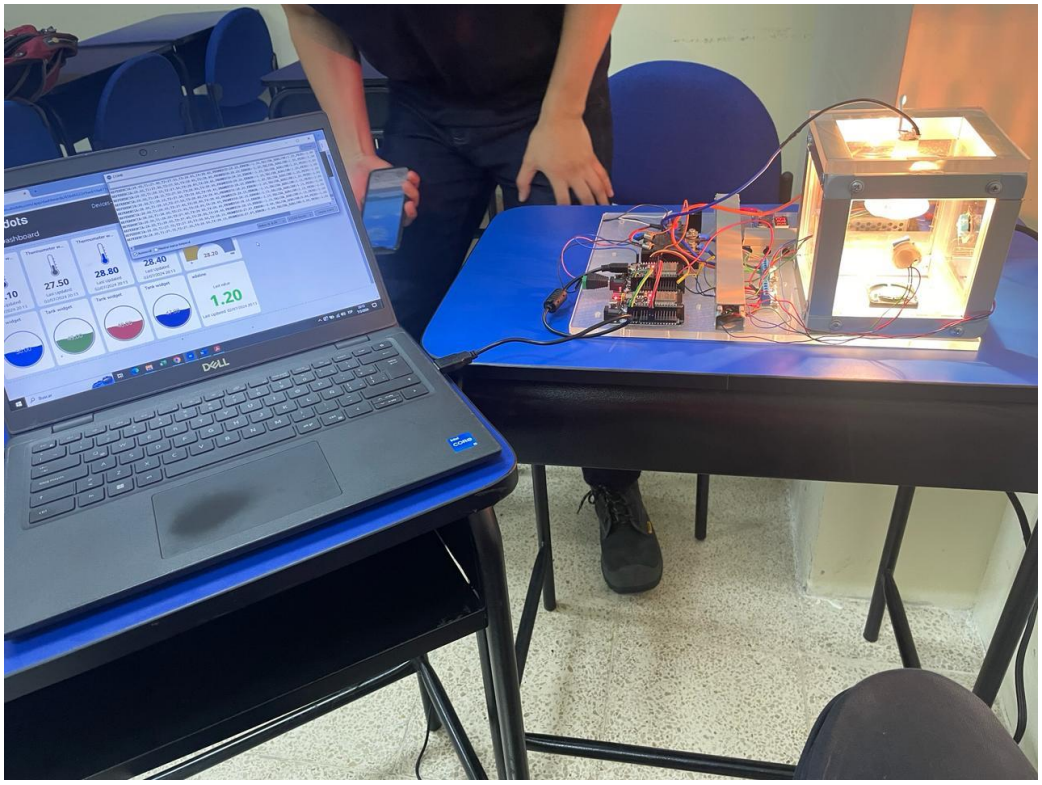


Figura 35: Monitoreo del Control de temperatura.

En la figura 35 se observa realizando pruebas cambiando el valor de referencia de temperatura.

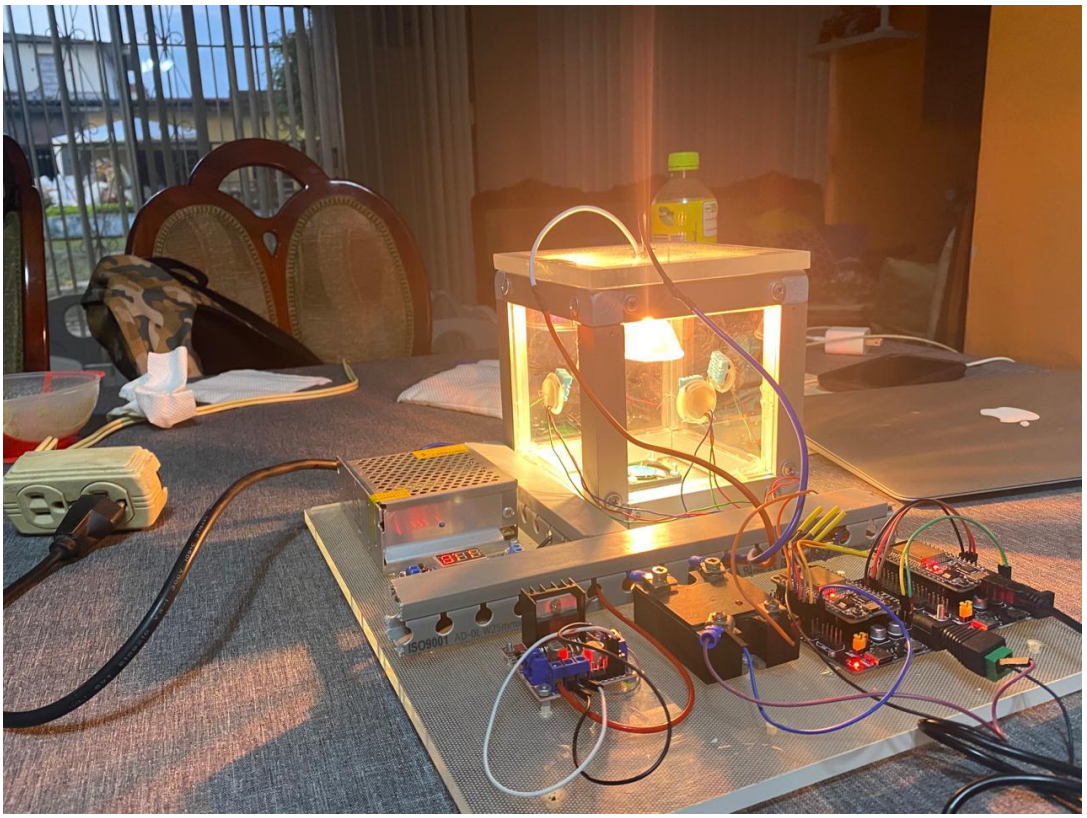


Figura 36: Ajuste de parámetros.

En la figura 36, se observa verificación de los datos.

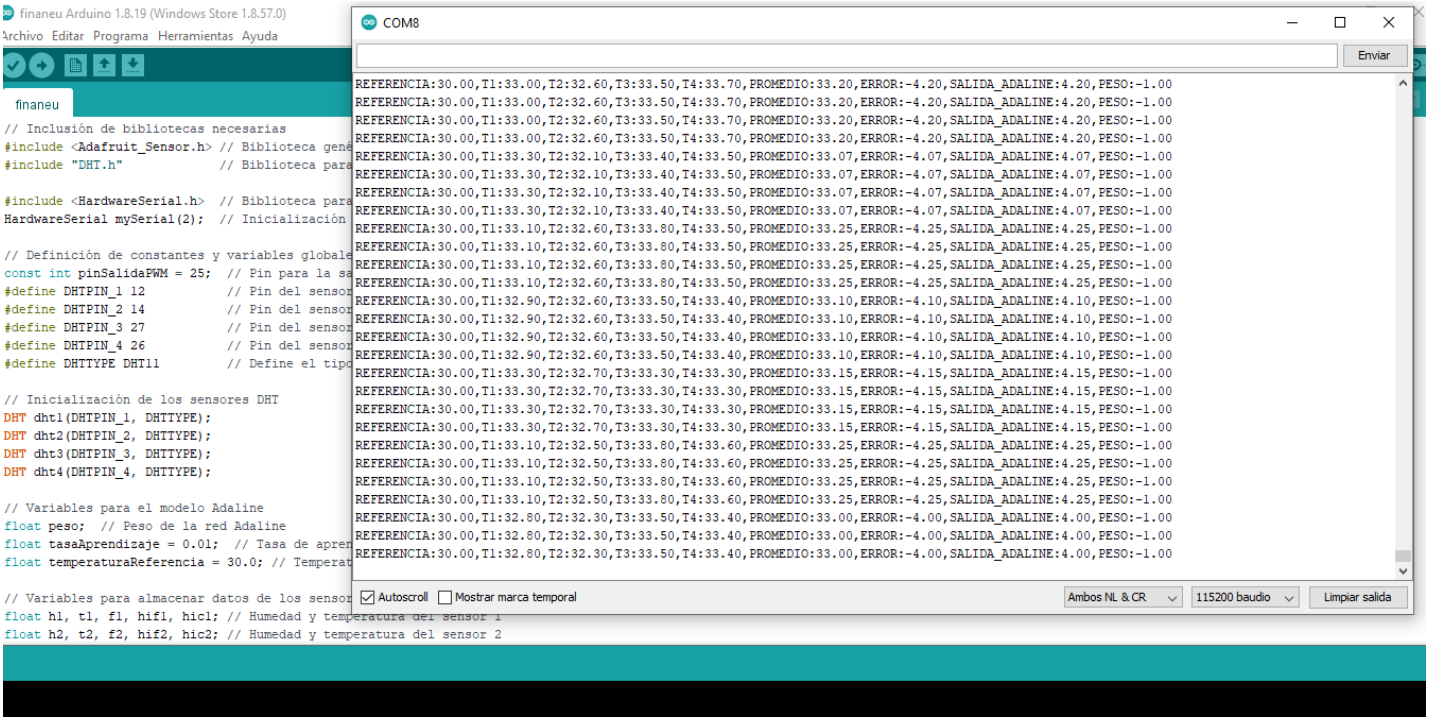


Figura 37: Monitoreo de valores de Temperatura.

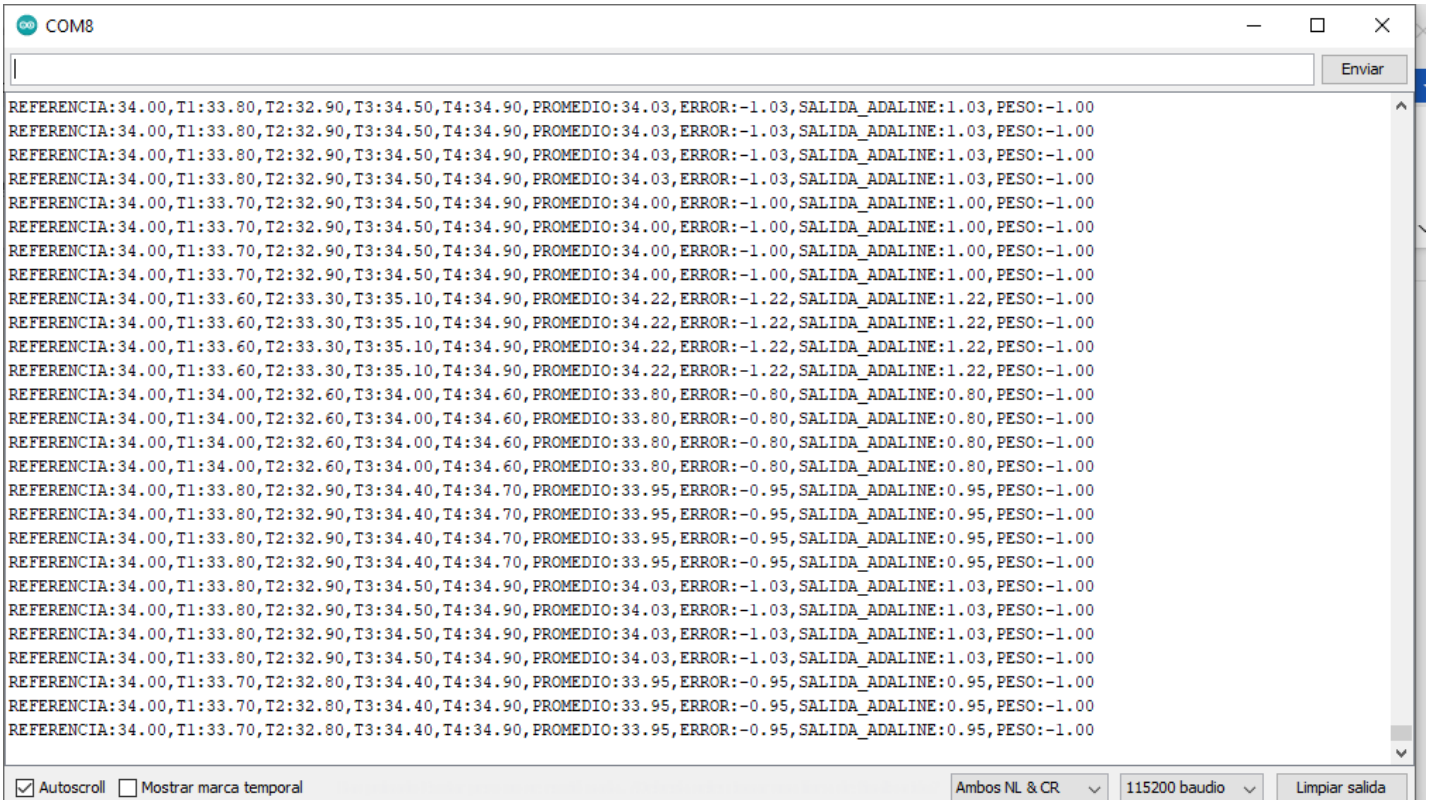


Figura 38: Valores de temperaturas según el valor de referencia.

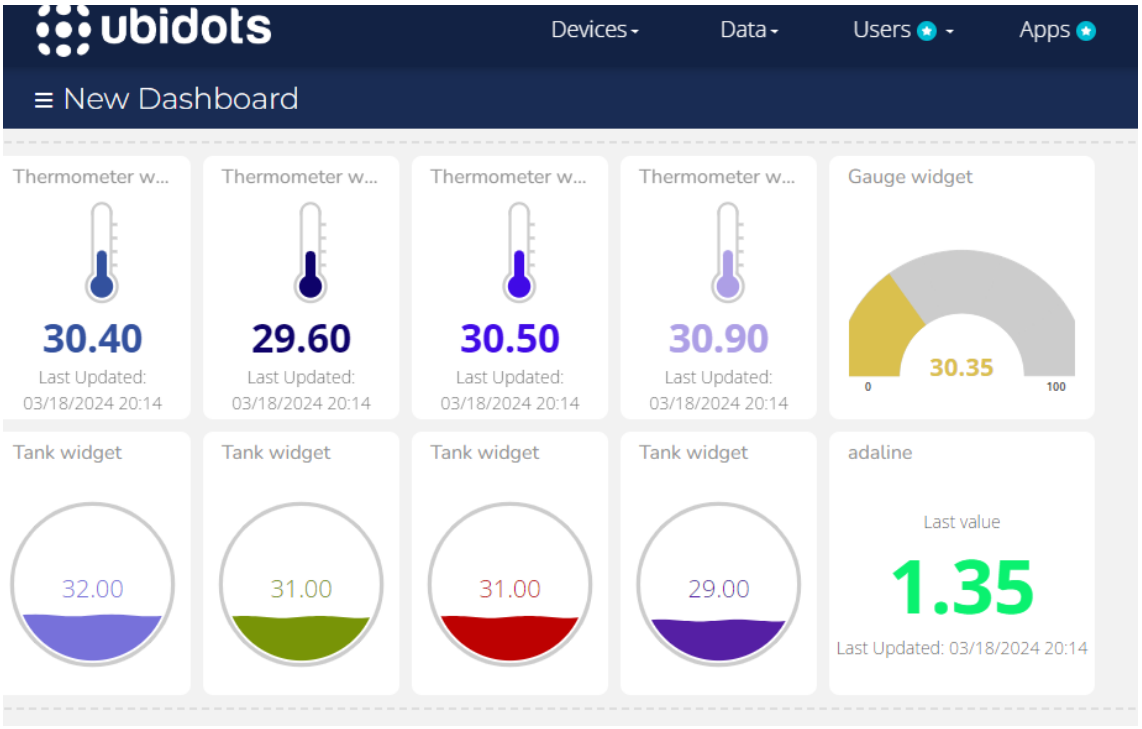


Figura 39: Valores de temperaturas a 30 grados centígrados en la interfaz de ubidots .

En la figura 39 se observan los valores de las temperaturas:

T1 Naranja, T2 Verde, T3 Amarillo, T4 Lila, Temperatura de referencia Azul, Temperatura promedio Gris, salida Adaline Negro, peso Azul, error Celeste.
Se observa que el valor promedio de temperaturas es muy cercano al de referencia, es decir 30 grados.

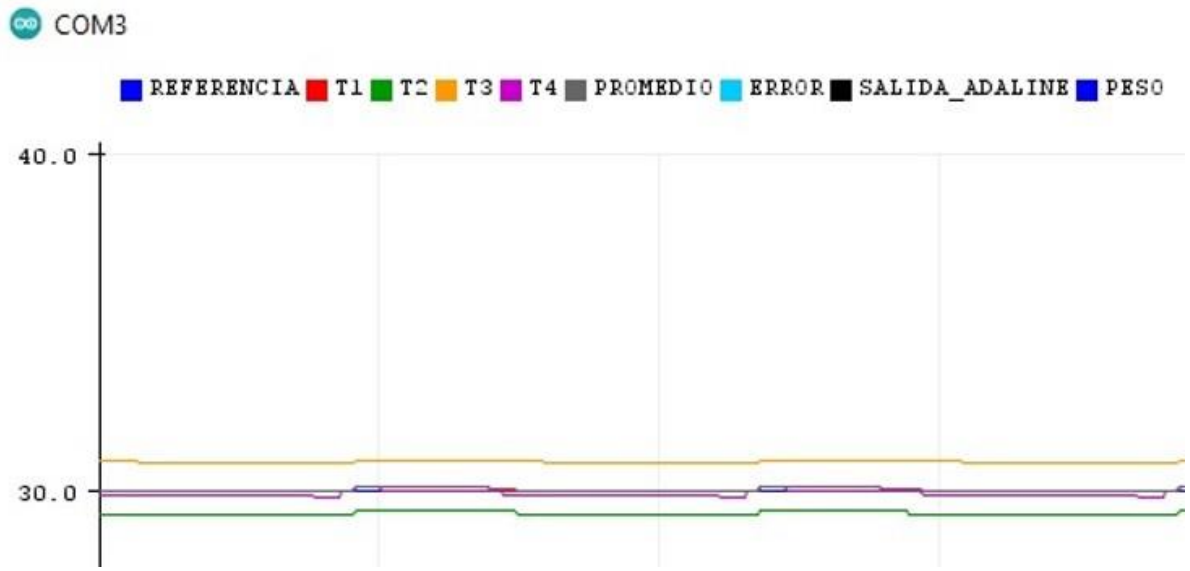


Figura 40: Grafico más cercano de las temperaturas a 30 grados centígrados.

Figura 41: Valores de temperaturas a 35 grados centígrados.

En la figura 41 se observan las variaciones de temperatura a una referencia de 35 grados centígrados. Como se nota todas las temperaturas tienden a los 35 grados.

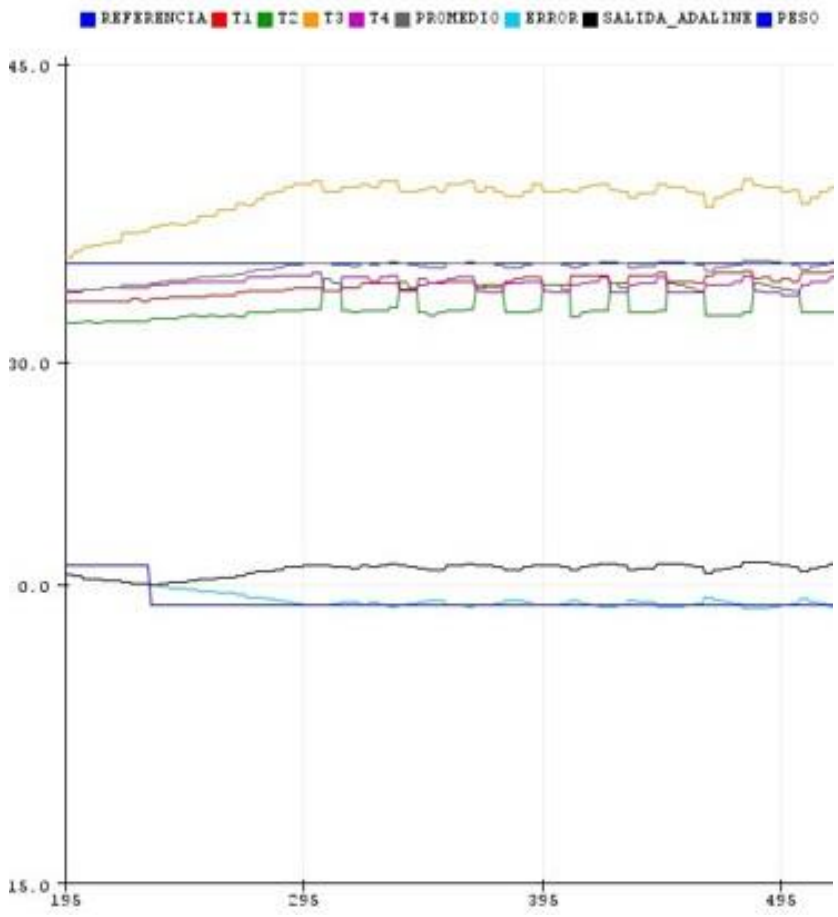


Figura 42: Grafico completo a 35 grados centígrados en la parte inferior se observa la salida del adaline, peso y error.

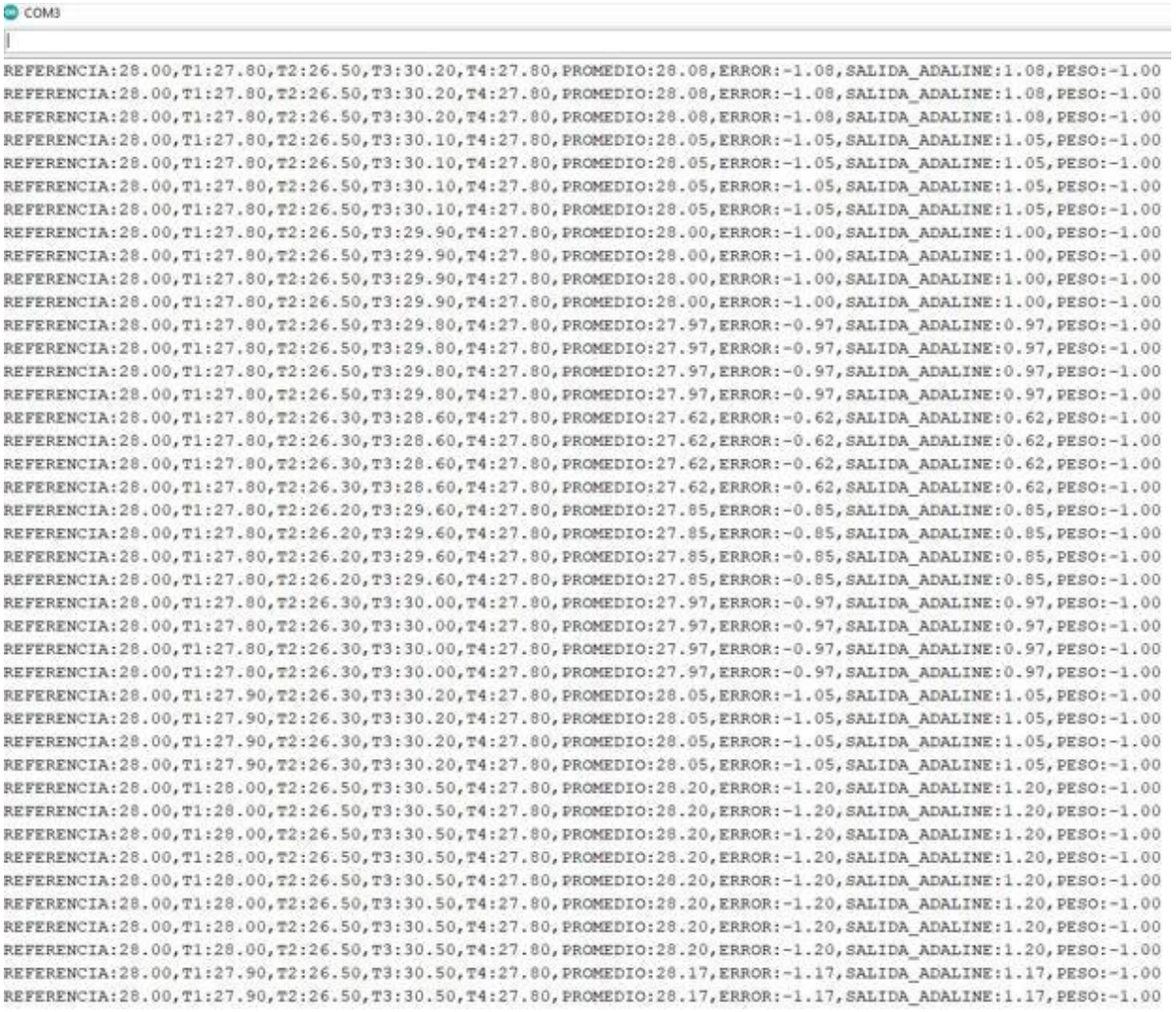


Figura 43: Datos de temperaturas visto en el monitor serie.

5. CONCLUSIONES

- Se realizó el programa de la red neuronal Adaline para el control y monitoreo de temperatura de una maqueta de galpón para pollos, utilizando el IDE de Arduino.

- Se realizó el diseño esquemático del proyecto con la utilización de sensores de temperatura – humedad, elementos de salida como el led de potencia y ventilador y como elemento de control - monitoreo se utilizaron dos microcontroladores ESP32.

- Se realizó a maqueta con las respectivas conexiones de sensores, actuadores e interfaces para el control detemperatura del criadero de pollos.

- Se presentaron los resultados en diferentes gráficas de las variables físicas del control neuronal, tanto en el PC como a distancia utilizando internet de las cosas (IoT), se muestra una llegada al valor de temperatura de referencia en un tiempo corto con una variación máxima de +/- 1 grado C.

6. RECOMENDACIONES

- Los cambios de lectura en el sistema en IoT en Ubidots son de aproximadamente 5 segundos. Esto se debe a que Ubidots depende de internet, lo que ocasiona un retardo entre la transmisión y la recepción de los datos.

- En las prácticas de control de temperatura, se pueden realizar cambios en el valor de referencia pero hay que considerar que, si se pone valores muy bajos como 20 grados, por ejemplo, el sistema no va a trabajar correctamente ya que no existe un sistema de enfriamiento.

- Para estudios posteriores con este proyecto se podría hacer comparaciones con otros tipos de controladores como el PID, Difuso, Predictivo, etc.

- Se debe tener cuidado con la ampliación de este proyecto utilizando la plataforma Ubidots, ya que después de cierto número de Widgets, el programa tiene costo.

7. BIBLIOGRAFIA

Cabrera, A. (2017). *DISEÑO Y SIMULACIÓN DE UN SISTEMA DE CONTROL DE TEMPERATURA E ILUMINACIÓN PARA LA CRIANZA DE POLLOS EN HACIENDAS USANDO UNA RED DE SENSORES.* Tesis de grado, UNIVERSIDAD DE GUAYAQUIL, ECUADOR.

(s.f.). *DESARROLLO DE APLICACIONES DE MONITOREO Y.*

Erazo, D., & Salgado, V. (2014). *DISEÑO E IMPLEMENTACIÓN DE UN SISTEMA DE CONTROL DE TEMPERATURA Y HUMEDAD PARA UN GALPÓN DE POLLOS DE LA AVÍCOLA "LA ESPERANZA".* Tesis de grado, UNIVERSIDAD TÉCNICA DEL NORTE, FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS CARRERA DE INGENIERÍA EN MECATRÓNICA, Ecuador.

Espinosa, B., & Orellana, M. (2021). *DESARROLLO DE APLICACIONES DE MONITOREO Y CONTROL BASADAS EN IOT A TRAVÉS DE LA PLATAFORMA UBIDOTS . APLICACIONES A SISTEMAS DE AUTOMATIZACIÓN BAJO ENTORNOS DE SIMULACIÓN.* Trabajo de titulación previo a la obtención del título de Ingeniero Electrónico, Cuenca.

Iraceburu, J., & Goicoechea, J. (2014).

Desarrollo e implementación de una red inalámbrica de sensores de temperatura y humedad . Trabajo Fin de Grado , Universidad Pública de Navarra, Pamplona.

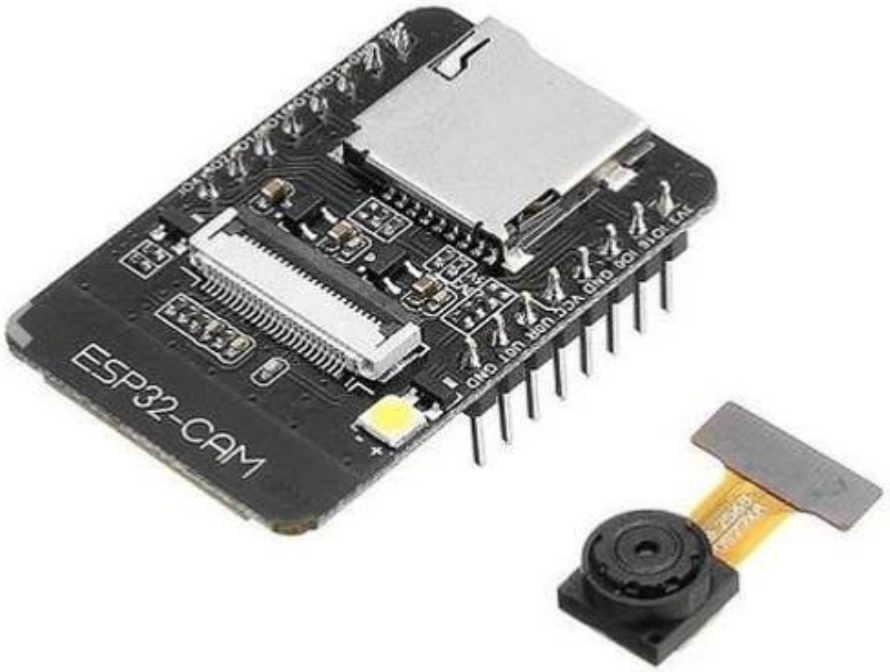
Protelec. (2023).

EL MÓDULO CONTROLADOR DE MOTORES L298N. WordPress.

Rodríguez, J. F. (2020). *Implementación de un control neuronal directo en el controlador lógico programable S7-1200 para aplicaciones industriales.* Grado en Ingeniería Mecatrónica, PAMPLONA.

8. ANEXOS

8.1 MICROCONTROLADOR ESP 32 CAM



DESCRIPCIÓN:

ESP32 CAM Modulo WiFi con Bluetooth y Camara OV2640 2MP, es una tarjeta de desarrollo que integra una pequeña cámara que puede funcionar de manera independiente.

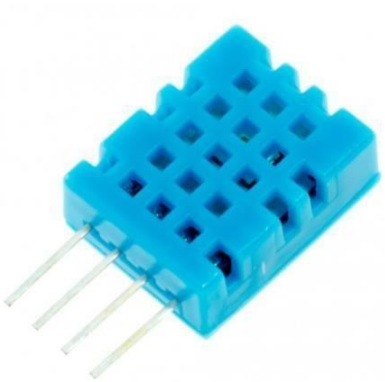
La camara OV2640 de 2MP integra un sensor de imagen CMOS UXGA (1632*1232) de 1/4 de pulgada. El pequeño tamaño del sensor y el bajo voltaje de operación brindan todas las características de una sola cámara UXGA y un procesador de imágenes. A través del control de bus SCCB, puede generar datos de imagen de 8/10 bits de varias resoluciones, como fotograma completo, submuestreo, zoom y ventanas.

La imagen UXGA de esta camara puede alcanzar hasta 15 cuadros por segundo (hasta 30 cuadros para SVGA y 60 cuadros para CIF). Los usuarios tienen un control completo sobre la calidad de la imagen, el formato de datos y la transmisión.

Especificaciones:

- Modelo: ESP32-CAM + Cámara OV2640
- Voltaje de Alimentación ESP 32 CAM: 5V
- Módulo Wi-Fi BT 802.11b/g/n
- Tipo de cámara: OV2640 2MP
- El modulo ESP 32CAM tiene CPU 32 bits de doble núcleo de baja potencia
- Frecuencia principal de hasta 240 MHz
- Potencia informática de hasta 600 DMIPS
- Velocidad de reloj de hasta 160 MHz
- Incorpora SRAM 520Kb, 4MPSRAM externa
- El modulo ESP-32 CAM Soporta interfaces: UART / SPI / I2C / PWM / ADC / DAC
- El modulo ESP-32-CAM Soporta cámaras OV2640 y OV7670, Flash Incorporado
- El modulo ESP 32-CAM Soporta tarjetas TF micro SD (Máximo 4 GB)
- El modulo ESP-32CAM Soporta la carga de imagen WiFi
- Compatible con modos de operación STA / AP / STA+AP
- Con antena PCB
- El modulo ESP32CAM integra conectores u.FL y FPC

8.2 SENSOR DE TEMPERATURA Y HUMEDAD DHT11



El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones académicas relacionadas al control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más.

Utilizar el sensor DHT11 con las plataformas Arduino/Raspberry Pi/Nodemcu es muy sencillo tanto a nivel de software como hardware. A nivel de software se dispone de librerías para Arduino con soporte para el protocolo "Single bus". En cuanto al hardware, solo es necesario conectar el pin VCC de alimentación a 3-5V, el pin GND a Tierra (0V) y el pin de datos a un pin digital en nuestro Arduino. Si se desea conectar varios sensores DHT11 a un mismo Arduino, cada sensor debe tener su propio pin de datos. Quizá la única desventaja del sensor es que sólo se puede obtener nuevos datos cada 2 segundos. Cada sensor es calibrado en fabrica para obtener unos coeficientes de calibración grabados en su memoria OTP, asegurando alta estabilidad y fiabilidad a lo largo del tiempo. El protocolo de comunicación entre el sensor y el microcontrolador emplea un único hilo o cable, la distancia máxima recomendable de longitud de cable es de 20m., de preferencia utilizar cable apantallado. Proteger el sensor de la luz directa del sol (radiación UV).

En comparación con el DHT22 y DHT21, este sensor es menos preciso, menos exacto y funciona en un rango más pequeño de temperatura / humedad, pero su empaque es más pequeño y de menor costo.

ESPECIFICACIONES TÉCNICAS

- Voltaje de Operación: 3V - 5V DC
- Rango de medición de temperatura: 0 a 50 °C
- Precisión de medición de temperatura: ±2.0 °C
- Resolución Temperatura: 0.1°C
- Rango de medición de humedad: 20% a 90% RH.
- Precisión de medición de humedad: 5% RH.
- Resolución Humedad: 1% RH
- Tiempo de sensado: 1 seg.
- Interface digital: Single-bus (bidireccional)
- Modelo: DHT11
- Dimensiones: 16*12*5 mm
- Peso: 1 gr.
- Carcasa de plástico celeste

PINES

- 1- Alimentación:+5V (VCC)
- 2- Datos (DATA)
- 3- No Usado (NC)
- 4- Tierra (GND)

*Recomendamos utilizar una resistencia de 4.7K Ohm en modo Pull-up, entre el pin de Datos y VCC

8.3 DRIVER L298

life.augmented

L298

Datasheet

Dual full-bridge driver

Multiwatt15 V

Multiwatt15 H

PowerSO-20

Product summary

L298

Product label

Features

- Operating supply voltage up to 46 V.
- Total dc current up to 4 A.
- Low saturation voltage.
- Overtemperature protection.
- Logical "0" input voltage up to 1.5 V (high noise immunity).

Applications

- Dual brush DC motors
- Stepper motors

Description

The L298 is an integrated monolithic circuit in a 15-lead multiwatt and PowerSO-20 packages. It is a high-voltage, high-current dual full-bridge driver designed to accept standard TTL logic levels and drive inductive loads such as relays, solenoids, DC and stepping motors.

Two enable inputs are provided to enable or disable the device independently of the input signals. The emitters of the lower transistors of each bridge are connected together and the corresponding external terminal can be used for the connection of an external sensing resistor. An additional supply input is provided so that the logic works at a lower voltage.

2 Absolute maximum ratings

Absolute maximum ratings are those values beyond which damage to the device may occur. These are stress ratings only and functional operation of the device at these conditions is not implied. Operating outside maximum recommended conditions for extended periods of time may impact product reliability and result in device failures.

Table 1. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_S	Power supply	50	V
V_{SS}	Logic supply voltage	7	V
V_i, V_{en}	Input and enable voltage	–0.3 to 7	V
I_O	Peak output current (each channel):		
	• Non repetitive (t = 100 ms)	3	A
	• repetitive (80% on –20% off; t_{on} =10 ms)	2.5	A
	• DC operation	2	A
V_{sens}	Sensing voltage	–1 to 2.3	V
P_{tot}	Total power dissipation (t_{case} = 75 °C)	25	W
T_{op}	Junction operating temperature	–25 to 130	°C
T_{stg}, T_j	Storage and junction temperature	–40 to 150	°C

Table 2. Thermal data

Symbol	Parameter		Power SO20	Multiwatt 15	Unit
$R_{th\ j\text{-}case}$	Thermal resistance junction-case	Max.	–	3	°C/W
$R_{th\ j\text{-}amb}$	Thermal resistance junction-ambient	Max.	13 (1)	35	°C/W

1. *Mounted on aluminum substrate*

3 Pin description

Figure 2. **Pin configuration**

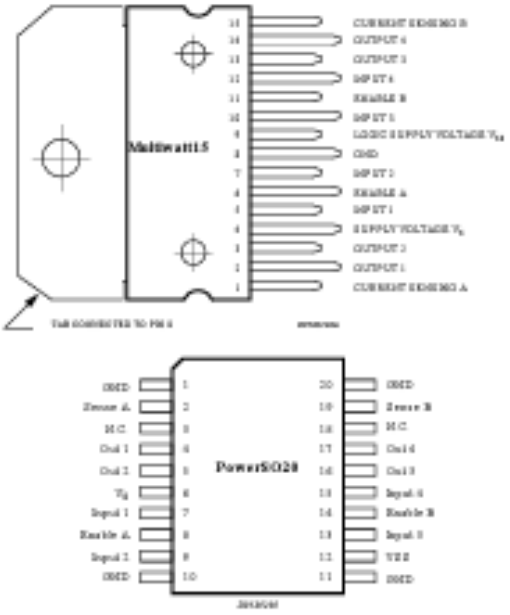


Table 3. **Pin function**

MW.15	Power SO	Name	Function
1, 15	2, 19	Sense A, Sense B	Between this pin and ground is connected the sense resistor to control the current of the load.
2, 3	4, 5	Out 1, Out 2	Outputs of the bridge A; the current that flows through the load connected between these two pins is monitored at pin 1.
4	6	V_S	Supply voltage for the power output stages. A non-inductive 100nF capacitor must be connected between this pin and ground.
5, 7	7, 9	Input 1, Input 2	TTL compatible inputs of the bridge A.
6, 11	8, 14	Enable A, Enable B	TTL compatible enable input: the L state disables the bridge A (enable A) and/or the bridge B (enable B).
8	1, 10, 11, 20	GND	Ground.
9	12	VSS	Supply voltage for the logic blocks. A 100nF capacitor must be connected between this pin and ground.
10, 12	13, 15	Input 3, Input 4	TTL compatible inputs of the bridge B.
13, 14	16, 17	Out 3, Out 4	Outputs of the bridge B. The current that flows through the load connected between these two pins is monitored at pin 15.
–	3, 18	N.C.	Not connected

8.4 RELE DE ESTADO SÓLIDO

Relés de Estado Sólido

Alta Fiabilidad, Entrada Vcc/Salida Vca, Entrada Vca/Salida Vca

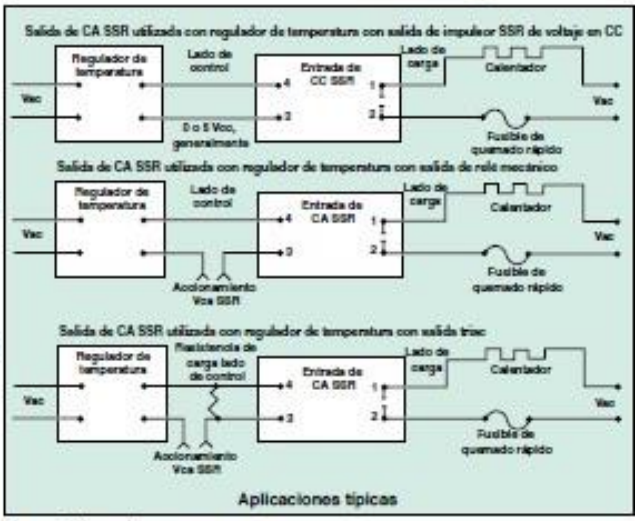
SSRL240AC10 se muestra en su tamaño real.

Serie SSRL

- Corriente máxima de hasta 100 A
- Múltiples millones de ciclos de vida
- Compatible con reguladores de temperatura
- Estado sólido, diseño SCR
- Conmutación por voltaje cero
- Líneas de control CA hasta 660 Vca
- Modelos de señal de control CA y CC
- Indicador de estado de entrada LED
- Zapata conductora térmica incluida

Los relés de la serie SSRL se utilizan para controlar calentadores de gran resistencia junto con reguladores de temperatura. Los relés de estado sólido son SPST, dispositivos conmutadores normalmente abiertos sin partes móviles, capaces de realizar millones de ciclos de operaciones. Aplicando una señal de control, un SSR enciende la corriente de carga CA, del mismo modo que los contactos móviles actúan en un contactor mecánico. Las cargas trifásicas pueden controlarse utilizando 2 o 3 SSR. Utilice 3 SSR para cargas trifásicas con conexión en "Y" o "estrella" utilizando una línea neutra. Dos SSR controlarán cargas "delta" sin ninguna línea neutra. Tres relés de estado sólido se utilizan también cuando no hay una carga neutra que proporcione redundancia y garantía de control adicional.

La "conmutación" tiene lugar en el punto de cambio de voltaje 0 del ciclo de corriente alterna. Por esta razón, no se genera ningún ruido electrónico apreciable, por lo que los SSR son ideales para entornos en los que hay aparatos susceptibles a RFI.



Aplicaciones típicas

Especificaciones comunes

Temperatura de funcionamiento: -20 a 80 °C (-5 a 175 °F)

Temperatura de almacenamiento: -40 a 80 °C (-40 a 175 °F)

Aislamiento: 4000 Vrms, de entrada a salida; 2500 Vrms de entrada/salida a tierra

Capacitancia: 8pF, de entrada a salida (máx.)

Rango de frecuencia: 47 a 63 Hz

Tiempo de encendido: 20 mseg., CA; 05 ciclos CC

Tiempo de apagado: 30 mseg., CA; 05 ciclos CC

Especificaciones de salida para modelos de entrada Vca y Vcc

Especificaciones	10 amperios	25 amperios	50 amperios	75 amperios	100 amperios
Máxima corriente en estado de encendido	10 A	25 A	50 A	75 A	100 A
Mínima corriente en estado de encendido	100 mA				
Máxima sobretensión en un ciclo	150 A	300 A	750 A	1000 A	1200 A
Máxima sobretensión en un segundo	30 A	75 A	150 A	225 A	300 A
I^2T (60 Hz), A^2s	416	937	2458	5000	6000

P-111

9. COSTOS DEL PROYECTO

	COSTOS DEL PROYECTO			
ITEM	EQUIPO	CANTIDAD	VALOR U	SUBTOTAL
1	Estructura	1	$ 120,00	$ 120,00
2	Sensores de temp. Y humedad DHT11	4	$ 12,00	$ 48,00
3	Relé de estado sólido 40 A	1	$ 20,00	$ 20,00
4	Led Potencia	1	$ 10,00	$ 10,00
5	Microcontrolador ESP32	2	$ 25,00	$ 50,00
6	Ventilador	1	$ 20,00	$ 20,00
7	Fuente de voltaje	1	$ 30,00	$ 30,00
8	Driver L298D	1	$ 25,00	$ 25,00
9	Varios (Canaletas,cabes,conectores,etc.)	Global	$ 90,00	$ 90,00
TOTAL				**$ 413,00**