INSTITUTO TECNOLÓGICO DE CD. GUZMÁN

MAESTRÍA EN INGENIERÍA ELECTRÓNICA

TESIS

TEMA:

CONTROL NEURONAL, Y MONITOREO REMOTO DE LA VARIABLE TEMPERATURA EN UN INVERNADERO

QUE PARA OBTENER EL TÍTULO DE:

MAESTRO EN INGENIERÍA ELECTRÓNICA

PRESENTA:

ING. EDGAR DAVID GUZMAN MARTINEZ

ASESOR:

M.I.P. JOSÉ DE JESÚS GARCÍA CORTÉS

COASESOR:

Dr. JAIME JALOMO CUEVAS

CD. GUZMÁN JALISCO, MÉXICO, JUNIO DE 2018

Cd. Guzmán, Jal. a 21/Junio/2018

Oficio No. S/N

ASUNTO : AUTORIZACIÓN DE IMPRESIÓN

ING. EDGAR DAVID GUZMAN MARTINEZ
N.C. M16290018

En cumplimiento con el documento normativo de las disposiciones para la operación de estudios de posgrado del Tecnológico Nacional de México y con base en la aprobación del Comité Tutorial comisionado para su revisión; la División de Estudios de Posgrado e Investigación le otorga la autorización de impresión de su trabajo de tesis intitulado:

"CONROL NEURONAL Y MONITOREO REMOTO DE LA VARIABLE TEMPERATURA EN UN INVERNADERO ",

dirigido por el **M.I.P. JOSÉ DE JESUS GARCÍA CORTÉS**, desarrollado como requisito parcial para la obtención del grado de Maestro en Ingeniería Electrónica, de acuerdo al plan de estudios MPIEO-2011-13.

Sin otro asunto en particular, quedo de usted.

S.E.P. TecNM
INSTITUTO TECNOLOGICO
DE CD. GUZMAN
DIVISION DE ESTUDIOS
DE POSGRADO E
INVESTIGACION

A T E N T A M E N T E

DR. HUMBERTO BRACAMONTES DEL TORO
JEFE DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.p. Archivo

DEDICATORIA

A mi familia que siempre estuvo a mi lado para apoyarme.

A Abisha, Melani, Gisel y Alejandra, que me brindaron sonrisas, alegrías y motivos para seguir adelante.

A mi madre Guadalupe Martínez que nunca dejo de creer en mí.

A aquellos que en el camino me ayudaron a superarme y mejorar.

“All the gods, all the heavens, all the hells, are within you.”- Joseph Campbell

AGRADECIMIENTOS

A Dios por guiarme.

A mi madre Guadalupe Martínez Quezada y padre Enrique Guzmán Contreras por nunca abandonarme.

A mi asesor el M.I.P. José de Jesús García Cortes por instruirme.

“Control Neuronal, y Monitoreo Remoto de la Variable Temperatura en un Invernadero”

RESUMEN

En este trabajo se presenta el control de temperatura dentro de un invernadero mediante el uso de una interfaz de monitoreo HMI SCADA y un control con redes neuronales para mantener una temperatura óptima para el crecimiento de las plantas.

La temperatura es uno de los factores que repercute directamente en el cultivo dentro de un invernadero debido a un mal control de esta puede acarrear consecuencias negativas al cultivo, generar un crecimiento desordenado de la hoja y mal crecimiento del fruto. En la región sur de Jalisco existe una gran cantidad de invernaderos que pueden llegar a necesitar un aumento en su producción y ahorro de sus recursos, y una forma de ello es conseguir un buen manejo de las variables encargadas del crecimiento de las plantas.

El sistema propuesto de control es un HMI SCADA asistido con redes neuronales para monitorear y controlar los cambios de temperatura dentro del invernadero, llevándose el control a cabo por un control de redes neuronales y un humificador AirWet para regular los cambios requeridos por el cultivo.

Palabras Clave

Invernadero, Temperatura, Redes neuronales, Control, HMI, AirWet, PLC.

"Neural Control and Remote Monitoring of the Temperature Variable in a Greenhouse"

ABSTRACT

The present project shows a temperature control within a greenhouse by using a HMI SCADA monitoring interface and a control with neural networks to maintain an optimum temperature for the growth of the plants.

The temperature is one of the factors that directly affects the crop in a greenhouse due to poor control of this can cause negative consequences to the crop; generate a messy growth of the leaf and poor growth of the fruit. In the southern region of Jalisco there is a large number of greenhouses that may need to increase their production and saving their resources, and one way is to get good management of the variables responsible for the growth of the plants.

The proposed control system is a SCADA HMI assisted with neural networks to monitor and control the temperature changes inside the greenhouse, taking control by a neural network control and an AirWet humidifier to regulate the changes required by the crop.

Keywords

Greenhouse, Temperature, Neural Networks, Control, HMI, AirWet, PLC

Índice de Contenido

Índice de figuras

Índice de tablas

Capítulo 1
Introducción

1 Introducción

1.1 Introducción: invernaderos en el sur de Jalisco

En la región sur de Jalisco existen una población considerable de invernaderos, y los cuales son una fuente de ingreso para esta región ya que proveen de trabajo a muchos habitantes del lugar, sin embargo, las condiciones en las que estos operan suelen llegar a ser rudimentarias o arbitrarias en lo que al manejo de las variables requeridas para el crecimiento de un cultivo se refiere, por esta razón en este proyecto se propone un control para regular la temperatura dentro del invernadero.

La temperatura en el invernadero es uno de los factores más importantes a tener en cuenta cuando de salud de cultivo se trata, pues aunque existen distintas variables que afectan el crecimiento y desarrollo del cultivo, la temperatura está ligada a varias de ellas como es el caso de la humedad relativa, la cual es inversamente proporcional a la temperatura. La modulación de esta variable dependerá del clima y región geográfica donde se encuentre el invernadero, por ejemplo pudiera ser necesario el uso de calentadores si el cultivo lo requiriera.

Usualmente la temperatura mínima por la mayoría de plantas oscila entre los 10° y 15° centígrados, mientras temperaturas mayores a 30° puede resultar dañina. En invernaderos relativamente grandes es recomendable establecer un monitoreo mediante sensores, pues realizarlo con personal puede resultar poco económico y no muy viable.

1.2 Antecedentes

La agricultura ha evolucionado con el paso de los años, las formas de cultivar y cosechar no son las mismas que en la década de los 50. La implementación de agricultura protegida en la actualidad ha permitido la producción de alimentos de manera considerablemente rápida en comparación con las formas tradicionales. La ventaja que ofrece un invernadero es la adaptación climática que provee para diferentes tipos de cultivo, siendo posible poder cultivar diferentes tipos de planta en un periodo similar de tiempo.

Los invernaderos tienen su origen en los países bajos alrededor de 1850, usados principalmente para el cultivo de uvas. Descubriendo así que al poder controlar el clima dentro del invernadero se obtenían mejores frutos ya que la temperatura constante permitía el crecimiento más rápido de la planta, además que poder cultivar frutos que solo crecen en climas cálidos.

Según datos publicados por la Secretaria de Agricultura, Ganadería, Desarrollo Rural y Alimentación (SAGARPA), en Jalisco existen diversos municipios que cuentan con invernaderos para el cultivo de hortalizas, teniendo un resultado de 3310.091 hectáreas ocupadas por agricultura protegida de las cuales, en el municipio de Zapotlán el Grande (Ciudad Guzmán) se encuentran 280.199, a su vez que el número de instalaciones que se encuentran en el municipio son 165 invernaderos y 35 macrotúneles, estos datos fueron publicados en el año 2016. [7]. En la región sur de Jalisco se cultiva en su mayoría aguacate, bayas y jitomate.

1.3 Objetivo

Modelar y controlar la variable temperatura en un invernadero utilizando un control inteligente con redes neuronales.

1.4 Objetivos específicos

- Caracterización del sistema a controlar.
- Definición de elementos de control a utilizar.
- Diseñar conexión de sensores de humedad.
- Diseñar diagrama conexión.
- Obtener el modelo matemático de la planta.
- Desarrollar Controlador PID con Matlab.
- Desarrollar Controlador PID con el PLC.
- Desarrollar el control con redes neuronales en Matlab.
- Análisis y comparación de resultados.

1.5 Definición del problema

La temperatura en un invernadero favorece el crecimiento de la planta y el desarrollo del fruto pero un mal manejo de ella puede llevar a que esta se marchite, su hoja crezca de maneja errónea o de frutos defectuosos, por ello es importante regular la temperatura sobre todo en verano, pues es cuando la temperatura tiende a elevarse, al menos en la región sur de Jalisco.

Las plantas son considerablemente susceptibles a los cambios de temperatura y las variables del tiempo. Es muy importante seleccionar especies que sean compatibles con el clima de la zona donde se vive y que no se afectadas por un violento cambio climático. Los cambios drásticos en la temperatura pueden actuar directamente modificando los procesos fisiológicos existentes, principalmente la fotosíntesis, e indirectamente, produciendo un patrón alterado del desarrollo subsecuente a la imposición del cambio ocurrido en la temperatura. Las plantas sólo pueden desarrollarse entre sus umbrales térmicos, o temperaturas mínimas y máximas, variando según la especie.

1.6 Justificación

El impacto de la temperatura en un invernadero es crucial pues la mayoría de los procesos biológicos se acelerarán con temperaturas altas, lo cual puede ser tanto positivo como negativo. Un rápido crecimiento o producción de frutos es un beneficio en la mayoría de los casos, sin embargo, la excesiva respiración que se produce es desfavorable porque implica que quedará menos energía disponible para el desarrollo de los frutos, resultando en unos frutos más pequeños. Algunos efectos se manifiestan a corto plazo mientras que otros lo harán a largo plazo.

La temperatura es la variable principal a controlar dentro de un invernadero. Esta variable es la que requiere de mayores sistemas mecánicos de actuación para su control, debido a que los sistemas de aireaciones son diseñados para el control de

esta variable, asimismo los sistemas mecánicos de calefacción y enfriamiento son utilizados para el mismo fin de control.

1.7 Hipótesis

El implementar un control inteligente para la temperatura nos permitirá obtener mejores condiciones dentro del invernadero, permitiendo su monitoreo y control remoto, con lo que se propone mejorar el rendimiento del cultivo a mediano y largo plazo, además de ofrecer una alternativa más accesible para su implementación debido a que se fabrica con componentes de bajo costo.

1.8 Metas

- Obtener el censado y registro de muestras de temperatura.
- Conocer el comportamiento de la planta.
- Control del AirWet con variador de velocidad.
- Implementar control PID con Matlab.
- Implementación del control con redes neuronales.

1.9 Limitaciones

- Costo elevado de los elementos de control (humificadore/ ventilador).
- Material disponible en la institución para realizar pruebas.
- PLC.

1.10 Delimitaciones

- Diseño de un transmisor y un receptor de HR por radiofrecuencia utilizando arduino.
- Diseño de un convertidor de voltaje-corriente para convertir de 1-5 V a 4-20mA para generar la salida de control al PLC.
- Instalación de sensores temperatura ds18b20 dentro del invernadero.
- Instalación variador de velocidad y sistema de humidificación dentro del invernadero.
- Diseño y aplicación de control PID en Rs logix 500.
- Diseño y aplicación de control PID en Matlab y LabVIEW.
- Diseño y aplicación de control con redes neuronales en Matlab y LabVIEW.

1.11 Contribución del trabajo a solución del problema

La implementación de un control basado en redes neuronales para el monitoreo y control en invernaderos de la temperatura, y de esta forma contribuir a un buen desarrollo del fruto en condiciones favorables.

Capítulo 2
Marco teórico

2 Marco teórico

2.1 Redes neuronales

Las redes neurales artificiales son aproximadores no lineales a la forma en que funciona el cerebro; por lo tanto no deben compararse directamente con el cerebro ni confundir los principios que fundamentan el funcionamiento de las redes neurales artificiales y el cerebro, ni pensar que las redes neurales se basan únicamente en las redes biológicas ya que sólo emulan en una parte muy simple el funcionamiento del cerebro humano. Además se debe considerar que las redes biológicas son generadoras de procesos neurobiológicos en que se establecen relaciones de complejidad muy alta, las cuales no se puede lograr con redes monocapas ni con redes multicapas.

Una neurona biológica es una célula especializada en procesar información. Está compuesta por el cuerpo de la célula (soma) y dos tipos de ramificaciones: el axón y las dendritas. La neurona recibe las señales (impulsos) de otras neuronas a través de sus dendritas y transmite señales generadas por el cuerpo de la célula a través del axón. La figura 2.1 muestra los elementos de una red neuronal natural [1].

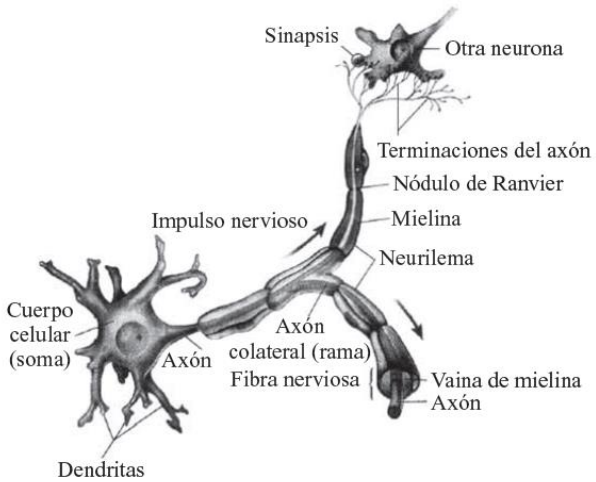


Figura 2.1 Elementos de una neurona biológica. [1]

Se calcula que en el cerebro humano existen aproximadamente 10^{15} conexiones de neuronas. Una de las características de las neuronas es su capacidad de comunicarse, las dendritas y el cuerpo celular reciben señales de entrada; el cuerpo celular las combina e integra y emite señales de salida. El axón transmite dichas señales a las terminales axónicos, los cuales distribuyen la información, se utilizan señales de dos tipos: eléctricas y químicas. La señal generada por la neurona y transportada a lo largo del axón es un impulso eléctrico, mientras que la señal que se transmite entre los terminales axónicos de una neurona y las dendritas de la otra es de origen químico. Para establecer una similitud directa entre la actividad sináptica y la analogía con las redes neurales artificiales podemos considerar que las señales que llegan a la sinapsis son las entradas a la neurona, éstas son ponderadas (atenuadas o simplificadas) a través de un parámetro denominado peso, asociado a la sinapsis correspondiente. Estas señales de entrada pueden excitar a la neurona (sinapsis con peso positivo) o inhibirla (peso negativo). El efecto es la suma de las entradas ponderadas. Si la suma es igual o mayor que el umbral de la neurona, entonces la neurona se activa (da salida). El mecanismo cambiante de todo o nada permite un ajuste de señales ya que las sinapsis son susceptibles a factores externos como fatiga o falta de oxígeno, pero este ajuste de señales también es un mecanismo de aprendizaje.

2.1.1 Redes neuronales artificiales

Las Redes Neuronales Artificiales (RNA) se definen como sistemas de mapeos no lineales cuya estructura se basa en principios observados en los sistemas nerviosos de humanos y animales. Constan de un número grande de procesadores simples ligados por conexiones con pesos. Las unidades de procesamiento se denominan neuronas.

Cada unidad recibe entradas de otros nodos y genera una salida simple escalar que depende de la información local disponible, guardada internamente o que llega a través de las conexiones con pesos.

Las neuronas artificiales simples fueron introducidas por McCulloch y Pitts en 1943. Una red neuronal se caracteriza por los siguientes elementos:

1. Un conjunto de unidades de procesamiento o neuronas.
2. Un estado de activación para cada unidad, equivalente a la salida de la unidad.
3. Conexiones entre las unidades, generalmente definidas por un peso que determina el efecto de una señal de entrada en la unidad.
4. Una regla de propagación, que determina la entrada efectiva de una unidad a partir de las entradas externas.
5. Una función de activación que actualiza el nuevo nivel de activación basándose en la entrada efectiva y la activación anterior.
6. Una entrada externa que corresponde a un término determinado como bias para cada unidad.
7. Un método para reunir la información, correspondiente a la regla del aprendizaje
8. Un ambiente en el que el sistema va a operar, con señales de entrada e incluso señales de error.

Donde en la mayoría de las redes se tiene una respuesta de la forma:

$$y = f\left(\sum_k \omega_k x_k\right) \quad , \qquad (2-1)$$

dónde:

- ω_k = *pesos de la liga de conexión*
- x_k = señales de salida de otros nodos o entradas externas
- $f(\bullet)$ = función no lineal simple

Usualmente la función puede ser sigmoidal, tangente hiperbólica, escalón entre otras.

Cada unidad recibe la entrada de otras unidades o de una fuente externa y procesa la información para obtener una salida que se propaga a las otras unidades. Una red puede tener una estructura arbitraria, pero las capas que contienen estas estructuras están definidas de acuerdo con su ubicación en la topología de la red neuronal. Las entradas externas son aplicadas en la primera capa, y las salidas se consideran la última capa. Las capas internas que no se observan como entradas o salidas se denominan capas ocultas. Por convención, las entradas no se consideran como capa porque no realizan procesamiento. La entrada total *u* de una unidad *k* es la suma de los pesos de las entradas conectadas, más un bias *θ*:

$$(2-2)$$

$$u = \sum_j \omega_j x_j + \theta$$

Si el peso ω_j es positivo se habla de una excitación y si el peso es negativo se considera una inhibición de la entrada. Si consideramos a las entradas como funciones del tiempo, la expresión anterior se convierte en:

$$(2-3)$$

$$u(t) = \sum_j \omega_j(t) x_j(t) + \theta(t)$$

2.1.2 Funciones de activación de las redes neuronales

La regla que logra establecer el efecto de la entrada total *u(t)* en la activación de la unidad *k* se denominafunción de activación *(Fk):*

$$(2-4)$$

$$y(t+1) = F_k\big(y(t), u(t)\big)$$

En muchas ocasiones esta función es de la forma no decreciente respecto a la entrada total de la unidad:

$$(2-5)$$

$$y(t+1) = F_k\left(\sum_j \omega_j(t)x_j(t) + \theta(t)\right)$$

Por lo tanto alguna de las funciones más utilizadas son:

- **Función escalón:** La función de activación escalón se asocia a neuronas binarias en las cuales, cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1; si es menor, la activación es 0. (Figura 2.2)

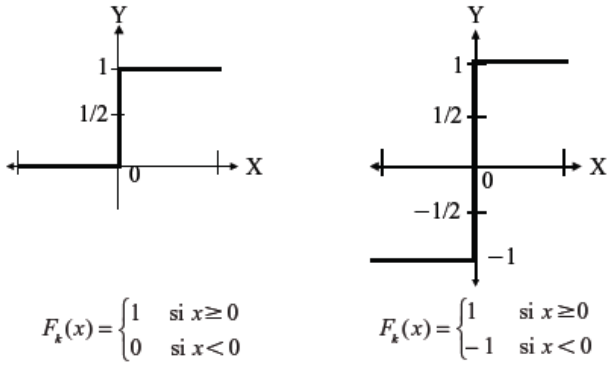


Figura 2.2 Función de activación escalón [1]

- **Función lineal y mixta:** La función lineal o identidad responde a la expresión $F_k(u) = u$. En las neuronas con función mixta, si la suma de las señales de entrada es menor que un límite inferior, la función se defi ne como 0 (o -1). Si dicha suma es mayor o igual que el límite superior, entonces la activación es 1. Si la suma de entrada está comprendida entre los dos límites, entonces la activación es 1. Si la suma de entrada está comprendida entre ambos límites, superior e inferior, entonces la activación se define como una función lineal de la suma de las señales de entrada (véase la figura 2.3).

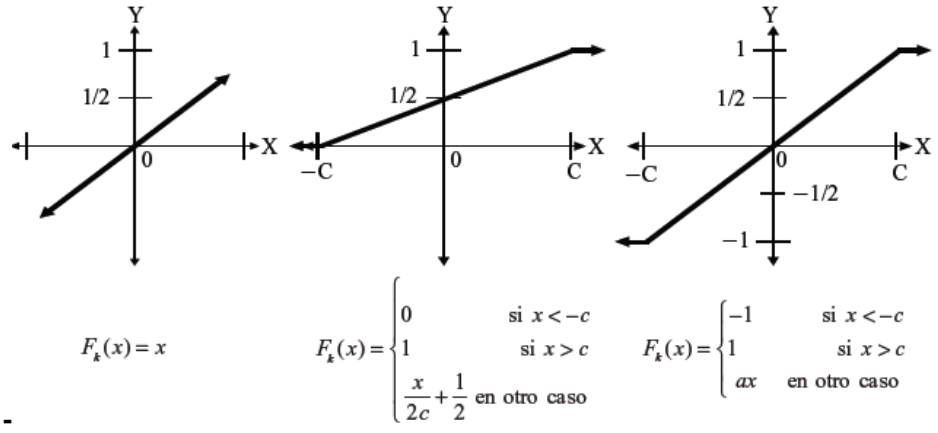


Figura 2.3 Función de activación lineal y mixta [1]

- **Función Sigmoidal:** Con la función sigmoidal el valor dado por la función es cercano a uno de los valores asintóticos. Esto hace que en la mayoría de los casos, el valor de salida esté comprendido en la zona alta o baja del sigmoide. De hecho, cuando la pendiente es elevada, esta función tiende a la función escalón. Sin embargo, la importancia de la función sigmoidal es que su derivada siempre es positiva y cercana a cero para los valores grandes positivos o negativos; además, toma su valor máximo cuando $x = 0$. Esto hace que se puedan utilizar reglas de aprendizaje definidas para las funciones escalón, con la ventaja, respecto a esta función, de que la derivada está definida en todo el intervalo (véase la fi gura 2.4).

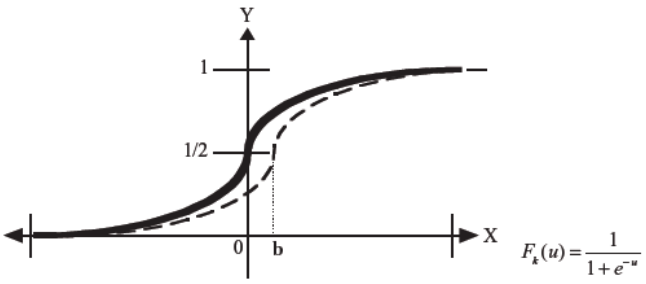


Figura 2.4 Función de activación Sigmoidal [1]

- **Función de Gauss:** Los mapeos ocultos algunas veces pueden realizarse con un solo nivel de neuronas mediante el uso de funciones de activación tipo Gauss, en lugar de funciones tipo sigmoidales. (Ver figura 2.5)

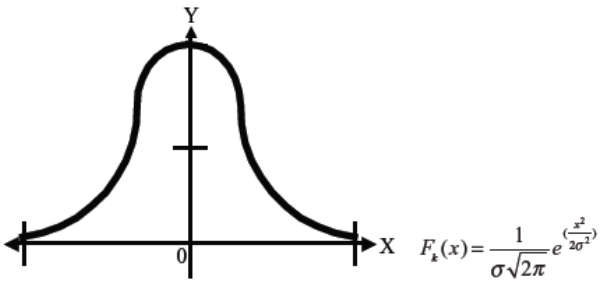


Figura 2.5 Función de Gauss [1]

2.1.3 Topología de las redes neuronales artificiales

Las topologías más utilizadas en las redes neuronales, de acuerdo con la forma de diferenciar las conexiones, son:

- Redes de propagación hacia adelante (feed forward): el flujo de información de las entradas a la salida fluye exclusivamente hacia adelante pasando por múltiples capas, pero no tiene retroalimentación.
- Redes recurrentes: contienen conexiones de retroalimentación lo que puede derivarse a un proceso de evolución hacia un estado estable en el que no haya cambio de activación de las neuronas.

Una RNA consta de un conjunto de elementos conectados entre sí y en los que se envía información a través de conexiones. En la figura 2.6 se muestra el esquema de una red Feed Forward debido al flujo de información que maneja.

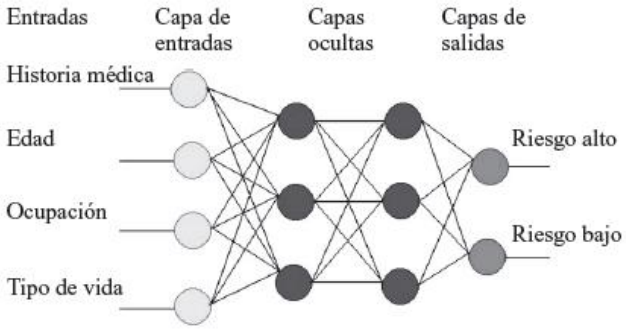


Figura 2.6 Esquema de RNA (feed forward) [1]

Los elementos básicos de una red neuronal son:

- Conjunto de unidades de procesamiento (neuronas)
- Conexiones entre unidades (asignando un peso o valor)
- Función de salida o activación para cada unidad de procesamiento

2.1.4 Entrenamiento de las redes neuronales

Se denomina entrenamiento al proceso de configuración de una red neuronal para que las entradas produzcan las salidas deseadas a través del fortalecimiento de las conexiones. Una forma de llevar esto a cabo es a partir del establecimiento de pesos conocidos con anterioridad, y otro método implica el uso de técnicas de retroalimentación y patrones de aprendizaje que cambian los pesos hasta encontrar los adecuados.

Además, el aprendizaje puede dividirse en supervisado o asociativo y no supervisado o auto-organizado. En el primer caso se introducen entradas que corresponden a determinadas salidas, ya sea por un agente externo o por el mismo sistema. En el segundo caso el entrenamiento se enfoca a encontrar características estadísticas entre agrupamientos de patrones en las entradas.

Uno de los métodos usados para el ajuste de los pesos es la hebbiana propuesto por Hen en 1949, este método propone que si dos entradas *j* y *k*, están

activas al mismo tiempo la conexión de las mismas debe ser fortalecida mediante la modificación del peso:

$$\Delta\omega_{jk} = \gamma y_j y_k \qquad (2-6)$$

donde γ es una constante de proporcionalidad positiva que representa la tasa de aprendizaje.

Otra regla utilizada es la regla delta, donde se ajustan los pesos a través de la diferencia actual y la deseada.

$$\Delta\omega_{jk} = \gamma y_j (d_k - y_k), \qquad (2-7)$$

donde d_k es la activación deseada.

2.2 Control PID

Un controlador PID (Proporcional Integrativo Derivativo) es un mecanismo de control genérico sobre una realimentación de bucle cerrado, ampliamente usado en la industria para el control de sistemas. El PID es un sistema al que le entra un error calculado a partir de la salida deseada menos la salida obtenida y su salida es utilizada como entrada en el sistema que queremos controlar. El controlador intenta minimizar el error ajustando la entrada del sistema.

El controlador PID viene determinado por tres parámetros: el proporcional, el integral y el derivativo. Dependiendo de la modalidad del controlador alguno de estos valores puede ser 0, por ejemplo un controlador Proporcional tendrá el integral y el derivativo a 0 y un controlador PI solo el derivativo será 0.

Cada uno de estos parámetros influye en mayor medida sobre alguna característica de la salida (tiempo de asentamiento, sobrepaso y error) pero también influye sobre las demás, por lo tanto por mucho que se ajuste no se encontraría un PID que redujera el tiempo de establecimiento a 0, el sobrepaso a 0, y el error a cero. Es por ello que se ajusta en un término medio en donde cumpla con las

características requeridas por el sistema. Los componentes del controlador aportan ciertas características al mismo siendo estas las siguietes:

- **Acción Proporcional (P):** La respuesta proporcional es la base de los tres modos de control, si los otros dos, control integral y control derivativo están presentes, éstos son sumados a la respuesta proporcional. “Proporcional” significa que el cambio presente en la salida del controlador es algún múltiplo del porcentaje del cambio en la medición. Este múltiplo es llamado “ganancia” del controlador. Para algunos controladores, la acción proporcional es ajustada por medio de tal ajuste de ganancia, mientras que para otros se usa una “banda proporcional”. Ambos tienen los mismos propósitos y efectos.
- **Acción Integral (I):** La acción integral da una respuesta proporcional a la integral del error. Esta acción elimina el error en régimen estacionario, provocado por el modo proporcional. Por contra, se obtiene un mayor tiempo de establecimiento, una respuesta más lenta y el periodo de oscilación es mayor que en el caso de la acción proporcional.
- **Acción Derivativa (D):** La acción derivativa da una respuesta proporcional a la derivada del error (velocidad de cambio del error). Añadiendo esta acción de control a las anteriores se disminuye el exceso de sobrepasos u oscilaciones.

2.2.1 Métodos de sintonización de un control PID mediante las reglas de Ziegler-Nichols

Ziegler y Nichols propusieron reglas para determinar los valores de la ganancia proporcional *Kp*, del tiempo integral *Ti* y del tiempo derivativo *Td*, basándose en las características de respuesta transitoria de una planta dada. Tal determinación de los parámetros de los controladores PID o sintonía de controladores PID la pueden realizar los ingenieros mediante experimentos sobre la planta. (Después de la propuesta inicial de Ziegler-Nichols han aparecido numerosas reglas de sintonía de controladores PID. Estas reglas están disponibles tanto en publicaciones técnicas como de los fabricantes de estos controladores). [2].

Hay dos métodos denominados reglas de sintonía de Ziegler-Nichols: el primero y el segundo método

Primer método (bucle abierto). En el primer método, la respuesta de la planta a una entrada escalón unitario se obtiene de manera experimental, tal como se muestra en la figura 2.7. Si la planta no contiene integradores ni polos dominantes complejos conjugados, la curva de respuesta escalón unitario puede tener forma de "S", como se observa en la figura 2.8. Este método se puede aplicar si la respuesta muestra una curva con forma de "S". Tales curvas de respuesta escalón se pueden generar experimentalmente o a partir de una simulación dinámica de la planta.

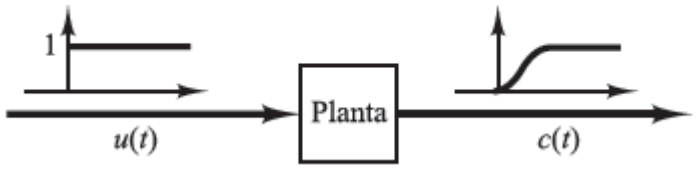


Figura 2.7 Respuesta a escalón unitario en una planta [2]

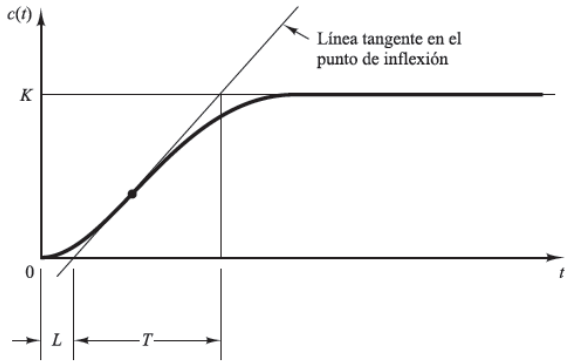


Figura 2.8 Curva de respuesta en forma de S [2]

La curva con forma de "S" se caracteriza por dos parámetros: el tiempo de retardo *L* y la constante de tiempo *T*. El tiempo de retardo y la constante de tiempo

se determinan dibujando una recta tangente en el punto de inflexión de la curva con forma de "S" y determinando las intersecciones de esta tangente con el eje del tiempo y con la línea *c(t)=K*, tal como se muestra en la figura 2.8. En este caso, la función de transferencia *C(s)/U(s)* se aproxima mediante un sistema de primer orden con un retardo del modo siguiente:

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1} \qquad (2-8)$$

Ziegler y Nichols sugirieron establecer los valores de *Kp*, *Ti* y *Td* de acuerdo con la fórmula que se muestra en la tabla I.

Tabla I Regla de sintonía de Zingler-Nichols basada en la respuesta escalón de una planta (primer método).

Tipo de controlador	K_p	T_i	T_d
P	$\frac{T}{L}$	∞	0
PI	$0.9\frac{T}{L}$	$\frac{L}{0.3}$	0
PID	$1.2\frac{T}{L}$	$2L$	$0.5L$

El controlador PID sintonizado mediante el primer método produce:

$$G_c(s) = K_p\left(1+\frac{1}{T_i s}+T_d s\right)$$
$$G_c(s) = 1.2\frac{T}{L}\left(1+\frac{1}{2Ls}+.5Ls\right) \qquad (2-9)$$
$$G_c(s) = \frac{0.6T\left(s+\frac{1}{L}\right)^2}{s}$$

Por tanto, el controlador PID tiene un polo en el origen y un cero doble en *s=.1/L*.

Segundo método (Bucle cerrado). En el segundo método, primero se fija *Ti*=∞ y *Td=0*. Usando sólo la acción de control proporcional (véase la figura 2.10), se incrementa *Kp* desde 0 hasta un valor crítico *Kcr,* en donde la salida presente oscilaciones sostenidas. (Si la salida no presenta oscilaciones sostenidas para cualquier valor que pueda tomar *Kp*, entonces este método no se puede aplicar.) Así, la ganancia crítica *Kcr* y el periodo *Pcr* correspondiente se determinan experimentalmente (véase la figura 2.11). Ziegler-Nichols sugirieron que se establecieran los valores de los parámetros *Kp*, *Ti* y *Td* de acuerdo con la fórmula que se muestra en la tabla II.

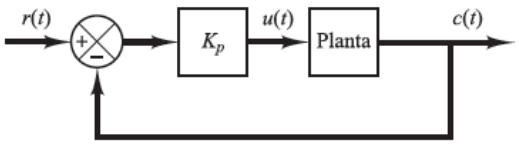


Figura 2.9 Sistema en lazo cerrado con control proporcional

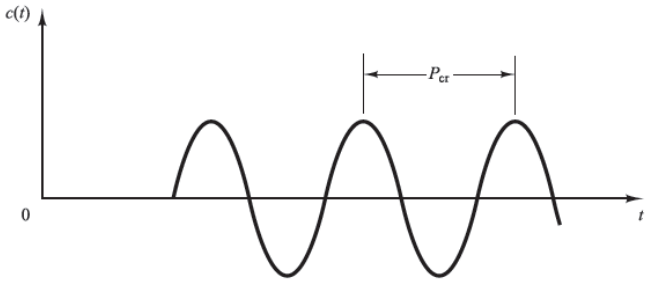


Figura 2.10 Oscilación sostenida con periodo Pcr (Pcr se mide en segundos)

Tabla II Regla de sintonía de Zingler-Nichols basada en la ganancia critica Kcr y periodo crítico Pcr (segundo método).

Tipo de controlador	K_p	T_i	T_d
P	$0.5K_{\text{cr}}$	∞	0
PI	$0.45K_{\text{cr}}$	$\frac{1}{1.2}P_{\text{cr}}$	0
PID	$0.6K_{\text{cr}}$	$0.5P_{\text{cr}}$	$0.125P_{\text{cr}}$

Obsérvese que el controlador PID sintonizado mediante el segundo método de las reglas de Ziegler-Nichols produce:

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

$$G_c(s) = 0.6K_{cr}\left(1 + \frac{1}{0.5P_{cr}s} + 0.125P_{cr}s\right) \qquad (2-10)$$

$$G_c(s) = \frac{0.075K_{cr}P_{cr}\left(s + \frac{4}{P_{cr}}\right)^2}{s}$$

Por tanto, el controlador PID tiene un polo en el origen y un cero doble en *s=-4/Pcr*.

Conviene darse cuenta de que, si el sistema tiene un modelo matemático conocido (como la función de transferencia), entonces se puede emplear el método del lugar de las raíces para encontrar la ganancia crítica *Kcr* y las frecuencias de las oscilaciones sostenidas ω_{cr}, donde $\frac{2\pi}{\omega_{cr}} = P_{cr}$. Estos valores se pueden determinar a partir de los puntos de cruce de las ramas del lugar de las raíces con el eje $j\omega$. (Obviamente, si las ramas del lugar de las raíces no cortan al eje $j\omega$ este método no se puede aplicar.)

2.3 Modulación por Ancho de Pulsos (PWM)

La modulación por ancho de pulsos (también conocida como PWM, siglas en inglés de pulse-width modulation) de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica (una senoidal o una cuadrada, ver figura 2.13), ya sea para transmitir información a través de un canal de comunicaciones o para controlar la cantidad de energía que se envía a una carga. Señales de PWM son utilizadas comúnmente en el control de aplicaciones. Su uso principal es el control de motores de corriente continua, aunque también pueden ser utilizadas para controlar válvulas, bombas, sistemas hidráulicos, y algunos otros dispositivos mecánicos. La frecuencia a la cual la señal de PWM se generará, dependerá de la aplicación y del tiempo de respuesta del sistema que está siendo controlado [9].

El ciclo de trabajo de una señal periódica es el ancho relativo de su parte positiva en relación con el período. Expresado matemáticamente:

$$(2-11)$$

$$D = \frac{\tau}{T} \ ,$$

dónde:

- D es el ancho de pulso
- τ es el tiempo donde la función es positiva (ancho de pulso)
- T es el periodo de la función

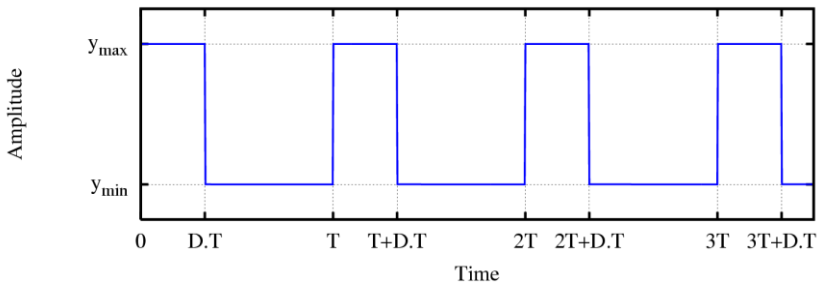


Figura 2.11 Ejemplo de PWM

La construcción típica de un circuito PWM se lleva a cabo mediante un comparador con dos entradas y una salida. Una de las entradas se conecta a un oscilador de onda dientes de sierra, mientras que la otra queda disponible para la señal moduladora. En la salida la frecuencia es generalmente igual a la de la señal dientes de sierra y el ciclo de trabajo está en función de la portadora.

La principal desventaja que presentan los circuitos PWM es la posibilidad de que haya interferencias generadas por radiofrecuencia. Estas pueden minimizarse ubicando el controlador cerca de la carga y realizando un filtrado de la fuente de alimentación.

Otra aplicación es enviar información de manera analógica. Es útil para comunicarse de forma analógica con sistemas digitales. Para un sistema digital, es relativamente fácil medir cuánto dura una onda cuadrada. Sin embargo, si no se tiene un conversor analógico digital no se puede obtener información de un valor analógico, ya que sólo se puede detectar si hay una determinada tensión, 0 o 5 voltios por ejemplo (valores digitales de 0 y 1), con una cierta tolerancia, pero no puede medirse un valor analógico. Sin embargo, el PWM en conjunción con un oscilador digital, un contador y una puerta AND como puerta de paso, podrían fácilmente implementar un ADC.

2.3.1 Señales Analógicas de Salida en Arduino PWM

$$D = PW = \frac{\tau}{T} \ , \qquad (2-11)$$

donde:

- PW es el ancho de pulso
- τ es el tiempo donde la función es positiva (ancho de pulso)
- T es el periodo de la función

La frecuencia se define como la cantidad de pulsos (estado on/off) por segundo y su expresión matemática es la inversa del periodo, como muestra la siguiente ecuación

$$(2-12)$$

$$f = \frac{1}{T} ,$$

donde:

- f es la frecuencia
- T es el periodo de la función

El periodo se mide en segundos, de este modo la unidad en la cual se mide la frecuencia (hertz) es la inversa a la unidad de tiempo (segundos). Existe otro parámetro asociado o que define a la señal PWM, denominado "Duty cycle", Ciclo de Trabajo, el cual determina el porcentaje de tiempo que el pulso (o voltaje aplicado) está en estado activo (on) durante un ciclo. Por ejemplo, si una señal tiene un periodo de 10 ms y sus pulsos son de ancho (PW) 2 ms, dicha señal tiene un ciclo de trabajo (duty cycle) de 20% (20% on y 80% off). La figura 2.12 muestra tres señales PWM con diferentes "duty cycles" [10].

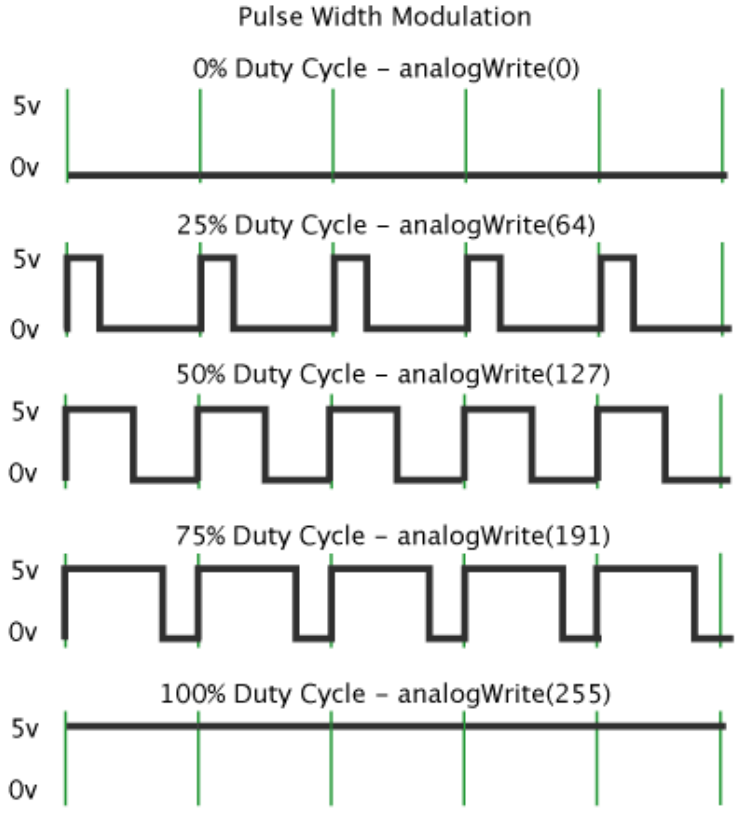


Figura 2.12 PWM con diferentes ciclos de trabajo

La señal PWM se utiliza como técnica para controlar circuitos analógicos. El periodo y el ciclo de trabajo (duty cycle) del tren de pulsos puede determinar la tensión entregada a dicho circuito. Si, por ejemplo, tenemos un voltaje de 5 V y lo modulamos con un duty cycle del 10%, obtenemos 0.5V de señal analógica de salida. Las señales PWM son comúnmente usadas para el control de velocidad de motores DC (si decrementas el ciclo de trabajo sobre la señal de control del circuito de potencia que actúa sobre el motor el motor se mueve más lentamente), ajustar la intensidad de brillo de un LED, etc.

En Arduino, con ATmega168 o ATmega328, la señal de salida PWM (pines 3,5,6,9,10 y 11) es una señal de frecuencia 490 Hz aproximadamente y que sólo nos permite cambiar el "duty cycle" o el tiempo que el pulso está activo (on) o inactivo (off), utilizando la función analogWrite(). Otra forma de generar señales PWM es utilizando la capacidad del microprocesador. La señal de salida obtenida de un microprocesador es una señal digital de 0 Voltios (LOW) y de 5 voltios (HIGH).

2.4 Convertidor Analógico Digital (ADC)

Los circuitos de conversión DC/AC tienen amplia aplicación en la industria. Son utilizados en variadores de velocidad, sistemas de alimentación ininterrumpida, filtros activos, etc. [11]. Un conversor o convertidor de señal analógica a digital (Conversor Analógico Digital, CAD; Analog-to-Digital Converter, ADC) es un dispositivo electrónico capaz de convertir una señal analógica, ya sea de tensión o corriente, en una señal digital mediante un cuantificador y codificándose en muchos casos en un código binario en particular. Donde un código es la representación unívoca de los elementos, en este caso, cada valor numérico binario hace corresponder a un solo valor de tensión o corriente.

En la cuantificación de la señal se produce pérdida de la información que no puede ser recuperada en el proceso inverso, es decir, en la conversión de señal digital a analógica y esto es debido a que se truncan los valores entre 2 niveles de cuantificación, mientras mayor cantidad de bits mayor resolución y por lo tanto menor información pérdida.

2.4.1 Exactitud y resolución de un ADC

Se define exactitud a la resolución que tendrá el ADC. Para ello primero se define el número máximo de bits de salida (la salida digital).

Este dato permite determinar el número máximo de combinaciones en la salida digital. Este número máximo está dado por: 2^n donde n es el número de bits (ver figura 2.13).

Figura 2.13 Diagrama de un convertidor ADC [12]

La resolución se define como el voltaje necesario (señal analógica) para lograr que en la salida (señal digital) haya un cambio del bit menos significativo (LSB). LSB significa: Least Significant Bit. Para hallar la resolución se utiliza la fórmula:

$$Res = \frac{ViFS}{2^n - 1}\ , \qquad (2-13)$$

donde:

- n: es el número de bits que tiene el convertidor analógico digital
- $ViFS$: es el voltaje que hay que poner a la entrada del convertidor ADC, para obtener una conversión máxima (todas las salidas serán iguales a “1”)

A continuación se muestran dos ejemplos de convertidores Analógico-Digital con diferente resolución.

Ejemplo no.1: Si se tiene un convertidor analógico – digital (ADC) de 4 bits y el rango de voltaje de entrada es de 0 a 15 voltios. Con n = 4 y ViFS = 15 Voltios (figura 2.14).

Figura 2.14 ADC de cuatro bits de resolución [12]

La resolución será: $Res = \frac{ViFS}{2^n - 1} = \frac{15}{16-1} = \frac{15}{15} = 1$ voltio / variación en el bit menos significativo.

Esto significa que un cambio de 1 voltio en la entrada, causará un cambio del bit menos significativo (LSB) a la salida. En este caso este bit es D0. Ver tabla III.

Tabla III Tabla de resolución de un ADC de cuatro bits

Entrada analógica	Salida digital de 4 bits			
Voltios	D3	D2	D1	D0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Ejemplo no.2: Un ADC de 8 bits genera sólo “1” (las 8 salidas en 1), cuando en la entrada hay un voltaje de 2.55 voltios (entrada analógica máxima). La resolución es $Res = \frac{ViFS}{2^n - 1} = \frac{2.55}{256-1} = \frac{2.55}{255} = 10\, mV/$ variacion en el bit menos significativo.

Se puede ver que mientras más bits tenga el convertidor, más exacta será la conversión. Si se tiene una señal de valor máximo de 15 voltios y aplicamos esta señal analógica analógica digital, se puede tener una idea de la variación de la resolución con el aumento del número de bits del convertidor, ver tabla IV.

Tabla IV Tabla de resoluciones de acuerdo con número de bits del ADC

# de bits del ADC	Resolución
4 bits	15 voltios / 15 = 1Voltio
8 bits	15 voltios / 255 = 58.8 miliVoltios
16 bits	15 voltios / 65536 = 0.23 milivoltios
32 bits	15 voltios / 4294967296 = 0.0000035 milivoltios

Esto significa que a mayor número de bits del ADC, un cambio más pequeño en la magnitud analógica causará un cambio en el bit menos significativo (LSB) de la salida, aumentando así la resolución [12].

2.5 Temperatura en un invernadero

La temperatura ideal en un invernadero varía en función del cultivo y sus estados, o etapas de desarrollo en las que se encuentre. Generalmente, la temperatura mínima requerida para las plantas de invernadero es de 10-15°C, mientras que 30°C podría ser la temperatura máxima.

Una variación o diferencia de temperatura de 5 – 7°C entre las temperaturas diurnas y nocturnas suele resultar beneficiosa para las plantas. La temperatura del suelo es incluso más importante que la temperatura del aire en un invernadero (Temperatura del suelo por debajo de 7°C, las raíces crecen más despacio y no absorben fácilmente el agua ni los nutrientes). Se debe conseguir un suelo templado, para que las semillas germinen y para se desarrollen los esquejes de raíces. La temperatura ideal para la germinación de la mayoría de las semillas es 18-25°C.

Resulta complicado regular las altas temperaturas en invernadero, sobre todo en verano. Por tanto, es conveniente disponer de un sistema de ventilación en la cubierta o contar con una malla de sombreo por fuera.

2.5.1 Influencia de las temperaturas críticas en el jitomate

Tabla V Temperaturas criticas de plantíos

	TOMATE	PIMIENTO	BERENJEMA	PEPINO	MELÓN	SANDÍA
Tª mínima letal	0-2	(-1)	0	(-1)	0-1	0
Tª mínima biológica	10-12	10-12	10-12	10-12	13-15	11-13
Tª óptima	13-16	16-18	17-22	18-18	18-21	17-20
Tª máxima biológica	21-27	23-27	22-27	20-25	25-30	23-28
Tª máxima letal	33-38	33-35	43-53	31-35	33-37	33-37

Las temperaturas críticas para un cultivo de jitomate rondan los 2°C y los 33°C, como se muestra en la tabla V. Para que un fruto logre tener su óptimo de cuaje, se deben presentar condiciones favorables de humedad y temperatura en el día y las noches, para lograr un óptimo cuaje la temperatura mínima diaria debe alcanzar entre 13 y 16°C, pero la temperatura óptima es de 21 a 25°C. Cuando se presentan temperaturas superiores a 32°C se complica la calidad de la polinización. Por tal razón, al final del ciclo (el paso del invierno a la temporada de calor) los frutos de la parte superior de la planta podrían no contar con los niveles de calidad que demanda el mercado. Es en esta época cuando el fruto se alarga, no llena bien y la consistencia disminuye.

Impacto del frío en el cultivo

Un cultivo cuya temperatura está por debajo del mínimo requerido (17 a 19°C), va sufrir los siguientes problemas fisiológicos:

- Se reduce la fertilidad del polen.
- El ovario se deforma.

- Es común observar racimos de frutos muy grandes que se bifurcan con el aspecto de manos.
- Los entrenudos se acortan.
- El nitrógeno en exceso tiene un efecto contrario al producido por el calor (plantas se reduce en crecimiento).
- Aumenta el número de tomate bofo o huecos.
- Aumenta el número de lacras en los frutos, si las temperaturas bajan de 5°C.
- En ciertas variedades aumenta el número de frutos deformes o con cara de gato.
- Si la temperatura baja a -2°C — el punto mínimo de resistencia — las plantas mueren, si ésta permanece por dos horas continuas.
- Las hojas de la planta se enrollan como medio de defensa contra las temperaturas extremas.

Impacto del calor en el cultivo

Cuando aumenta la temperatura por encima de 35°C se disminuye el número de granos de polen. Cualquier floración en esos días disminuye drásticamente, su cuaje o polinización y esos racimos se pierden o se obtienen sólo dos o tres tomates. Además:

- Las flores en los racimos disminuyen, es muy común apreciar racimos muy raquíticos.
- El tubo polínico, se alarga y por tal razón es imposible una fecundación natural.
- El número de frutos bofos o huecos aumenta y la calidad en los mismos baja drásticamente.
- En el caso de los tomates Roma (figura 2.15), se nota un alargamiento del fruto, se pierde su forma típica y en ocasiones se ponen puntiagudos.

- Los tallos se alargan. El primer racimo sale muy alto y los espacios entre un racimo y otro son muy largos.
- Cuando se usa acolchado, deben monitorear las temperaturas del cuello de la raíz y no usar más agua de la necesaria, ya que las enfermedades fungosas en el sistema radicular aumentan en estas condiciones.

Figura 2.15 Tomate Roma del invernadero del ITCG

2.6 Tipos de invernadero

Un invernadero es una construcción agrícola de estructura metálica, usada para el cultivo y/o protección de plantas, con cubierta de película plástica traslucida que no permite el paso de la lluvia al interior y que tiene por objetivo reproducir o simular las condiciones climáticas más adecuadas para el crecimiento y desarrollo de las plantas cultivadas establecidas en su interior, con cierta independencia del medio exterior y cuyas dimensiones posibilitan el trabajo de las personas en el interior. Los

invernaderos pueden contar con un cerramiento total de plástico en la parte superior y malla en los laterales.

Uno de los rasgos más importantes en un invernadero es la ventilación y para que se dé una ventilación efectiva, es recomendable que el área de ventilas sea aproximadamente igual del 15% al 30% del área del piso ocupado por la nave de invernadero. El nivel de enfriamiento es mejorado cuando las cortinas de las paredes laterales son incluidas en el área total de ventilación, ver figura 2.16.

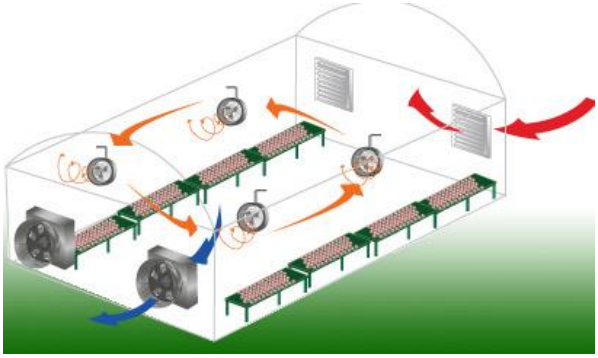


Figura 2.16 Ventilación sugerida para un invernadero [13]

Las características y formas del invernadero estarán dispuesta por las condiciones climáticas (temperatura, luz solar, lluvia y aire) y orografía, conforme a lo mencionado se establece la orientación de la estructura, ver figura 2.17.

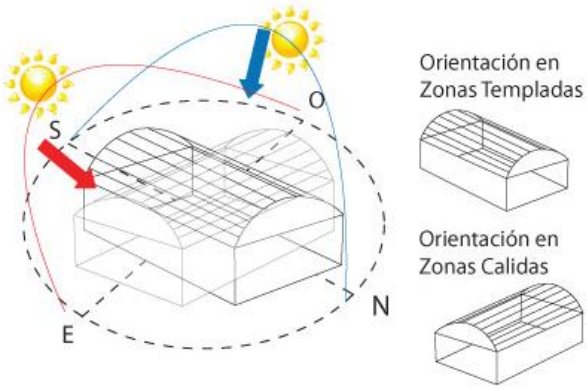


Figura 2.17 Orientación de la estructura según el clima [13]

Debido a esto puede intentarse una clasificación según criterios (por ejemplo materiales para la construcción, tipo de material de cobertura, características de la techumbre, etc.). Los invernaderos pueden clasificarse en:

- Invernadero-túnel.
- Invernadero capilla (a dos aguas).
- Invernaderos en diente de sierra.
- Invernadero con techumbre curva.
- Invernadero tipo “parral” ó “almeriense”.
- Invernadero “holandés” (tipo Venlo).

2.6.1 Descripción de los diferentes tipos de invernaderos

a) Invernadero-túnel

Es difícil establecer una línea divisoria entre lo que es un invernadero y un macro túnel, por no existir un parámetro definido, en México gracias a la norma mexicana que da definido como invernadero (ver figura 2.18). En general, de acuerdo a diferentes opiniones al respecto, podemos definir como invernadero aquella estructura que supera los 2,75-3,00 m3/m2.

Ventajas:

- Alta resistencia a los vientos y de fácil instalación.
- Tiene un alto grado de paso de luz solar.
- Apto tanto para materiales de cobertura flexible como rígidos

Desventajas:

- Relativamente pequeño volumen de aire retenido (escasa inercia térmica) pudiendo ocurrir el fenómeno de inversión térmica.
- Solamente recomendado para cultivos de bajo a mediano porte (lechuga, flores, frutillas, etc).

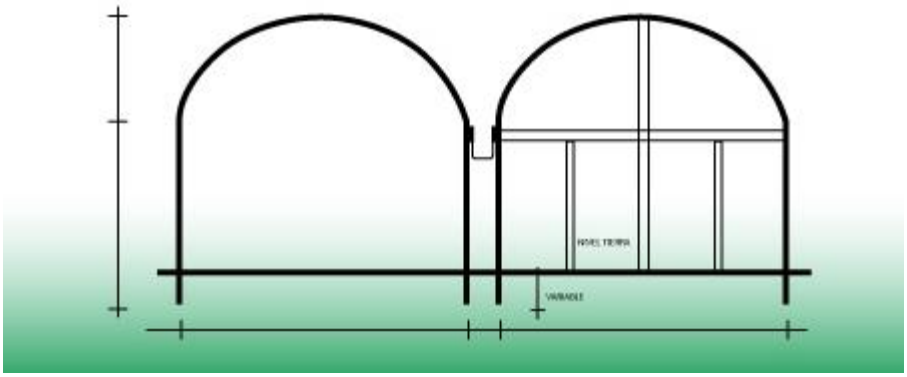


Figura 2.18 Forma de invernadero túnel [13]

b) Invernadero capilla (a dos aguas).

Se trata de una de las estructuras más antiguas, empleadas en el forzado. La pendiente del techo (cabio) es variable según la radiación y pluviometría variando normalmente entre 15° y 35°, ver figura 2.19. Las dimensiones del ancho, varían entre 6 y 12 m (incluso mayores), por largo variable. Las alturas de los laterales varían entre 2,0-2,5 m y la de cumbrera 3,0-3,5 m.

La ventilación de estos invernaderos en unidades sueltas, no ofrece dificultades; solo se hace difícil cuando varios de estos invernaderos se agrupan formando baterías.

Ventajas:

- Construcción de media a baja complejidad.
- Utiliza materiales de bajo costo dependiendo de la zona (postes de madera pinos, eucaliptos, etc).
- Apto tanto para materiales de cobertura flexible como rígidos

Desventajas:

- Problema de ventilación con invernaderos en batería.
- Misma altura cenital, tiene menor volumen encerrado que invernaderos curvos.

- Mayores números de elementos que disminuyen la transmisión de luz solar.
- Elementos de soportes internos que dificultan los desplazamientos y el emplazamiento de cultivos.

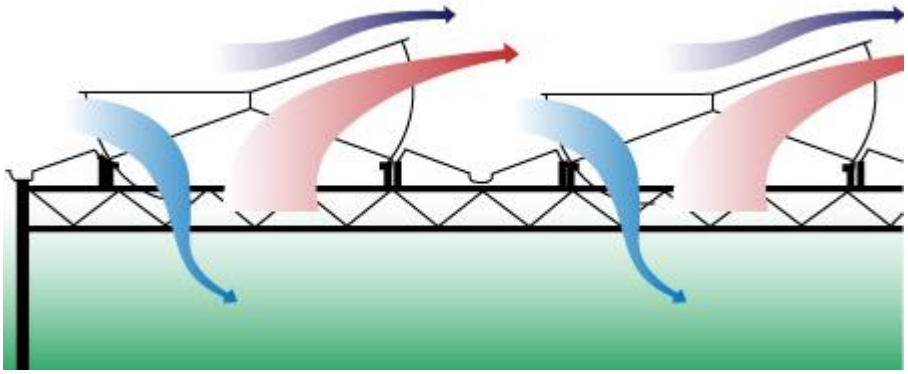


Figura 2.19 Forma de invernadero doble capilla [13]

c) Invernaderos en diente de sierra.

Una variación de los invernaderos tipo capilla, que se comenzó a utilizar en zonas con muy baja precipitación y altos niveles de radiación, fueron los invernaderos a una vertiente. Estos invernaderos, contaban con una techumbre única inclinada en ángulos que variaban entre 5° y 15° (orientados en sentido este-oeste y con presentación del techo hacia la posición del sol - norte para el hemisferio sur-).
El acoplamiento lateral de este tipo de invernaderos da origen a los conocidos “dientes de sierra” (figura 2.20). La necesidad de evacuar el agua de precipitación, determinó una inclinación en las zonas de recogida desde la mitad hacia ambos extremos.

Ventajas:

- Construcción de complejidad.
- Excelente ventilación.
- Empleo de materiales de bajo costo

Desventajas:

- Sombreo resulta mayor que en capilla debido al número de elementos.

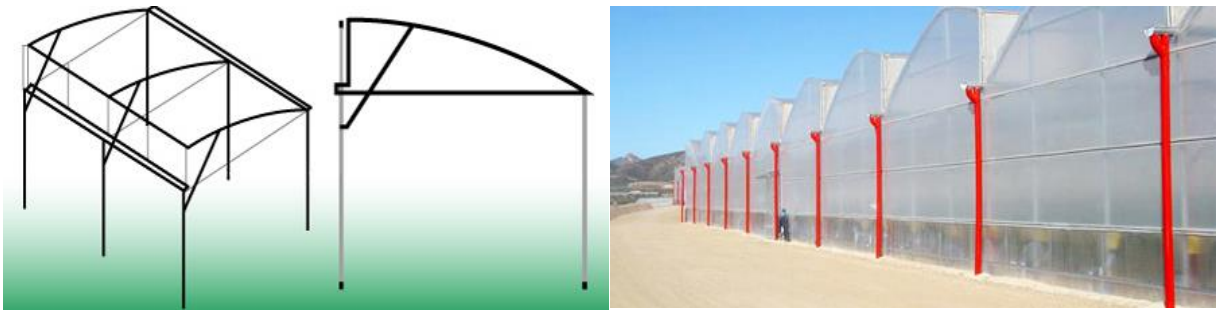


Figura 2.20 Forma de un invernadero diente de sierra [13]

El invernadero instalado en el Instituto Tecnológico de Cd. Guzmán es de este tipo, ver figura 2.21.

Figura 2.21 Invernadero diente de sierra del ITCG

d) Invernadero con techumbre curva.

Este tipo de invernaderos, tienen su origen en los invernaderos-túneles. Por lo común, son de tipo metálicos (caños de 2”a 2,5” de diámetro o bien perfiles triangulares con hierro redondo trefilado de 8-10 mm de diámetro) o bien con techumbres metálicas y postes de madera. Dentro de este tipo de invernaderos,

pueden encontrarse diferentes alternativas según la forma que adopta el techo (circulares, semi-elípticos o de medio punto, ojivales, etc.). Las dimensiones más comunes de estos invernaderos van de 6,0-8,0 m de ancho por largo variable. En la zona del cinturón hortícola de la ciudad de Santa Fe, existe una alternativa de muy bajo costo (más próxima al tipo semi-elíptico) construida con postes de madera y techumbre de madera arqueada o caña. Se trata de estructura muy endeble y de baja altura, tornándose en una importante limitante para el clima de la zona.

Ventajas:

- Estructuras con pocos obstáculos en su estructura.
- Buena ventilación.
- Buena estanqueidad a la lluvia y al aire.
- Permite la instalación de ventilación cenital a sotavento y facilita su accionamiento mecanizado.
- Buen reparto de la luminosidad en el interior del invernadero.
- Fácil instalación

Desventajas:

- Elevado coste.
- No aprovecha el agua de lluvia

e) Invernadero tipo “parral” ó “almeriense”.

Son invernaderos originados en la provincia de Almería (España), de palos y alambres, denominados “parral” por ser una versión modificada de las estructuras o tendidos de alambre empleados en los parrales para uva de mesa. Actualmente existe una versión moderna a los originales, que se construyen con caños galvanizados como sostenes interiores, permaneciendo el uso de postes para los laterales de tensión o aún, siendo reemplazados también éstos por muertos enterrados, para sujeción de los vientos constituidos por doble alambre del 8. Estos

invernaderos suelen tener una altura en la cumbrera de 3,0 a 3,5 m, la anchura variable, pudiendo oscilar en 20 metros o más por largo variable. La pendiente es casi inexistente, o bien (en zonas con pluviometría de riesgo) suele darse 10°-15°, lo que represente altura de los laterales del orden de 2,0-2,3 m. Se ventila solamente a través de las aberturas laterales. En la techumbre sólo se utiliza un doble entramado de alambre, por entre el cual se coloca la lámina de polietileno, sino otra sujeción.

Ventajas:

- Es económica su construcción.
- Gran adaptabilidad para la geometría del terreno.
- Mayor resistencia al viento.
- Aprovecha el agua de lluvia en periodos secos.
- Presenta uniformidad luminosa

Desventajas:

- Poca existencia de aire.
- Mala ventilación.
- La instalación de ventanas cenitales resulta difícil.
- Envejece rápido la estructura
- Difícil mecanización para las labores de cultivo.

f) Invernadero “holandés” (tipo Venlo).

Son invernaderos de vidrio, los paneles descansan sobre los canales de recogida del agua pluvial. La anchura de cada módulo es de 3,2 m y la separación entre postes en el sentido longitudinal es de 3 m.

Estos invernaderos carecen de ventanas laterales (puede ser debido a que en Holanda no existen demasiadas exigencias en cuanto a ventilación) [13]. En vez,

tiene ventanas cenitales, alternadas en su apertura (una hacia un lado y la siguiente hacia el otro) cuyas dimensiones son de 1,5 m de largo por 0,8 m de ancho.

Ventajas:

- Buena estanqueidad lo que facilita una mejor climatización de los invernaderos

Desventajas:

- La abundancia de elementos estructurales implica una menor transmisión de luz.
- Su elevado coste.
- Naves muy pequeñas debido a la complejidad de su estructura

2.7 Controlador Lógico Programable (PLC)

Un controlador lógico programable o PLC por sus siglas en ingles son dispositivos electrónicos que pueden ser programados para el control de procesos, a diferencia de las computadoras los PLC están diseñados para múltiples señales de entrada y de salida, rangos de temperatura ampliados, inmunidad al ruido eléctrico y resistencia a la vibración y al impacto. Los programas para el control de funcionamiento de la máquina se suelen almacenar en baterías copia de seguridad o en memorias no volátiles. Un PLC es un ejemplo de un sistema de tiempo real "duro", donde los resultados de salida deben ser producidos en respuesta a las condiciones de entrada dentro de un tiempo limitado. Sin embargo el PLC ha evolucionado y ahora permite mayor variedad de control debido a su versatilidad.

El PLC fue desarrollado inicialmente por un grupo de ingenieros de General Motors en 1968, donde se formuló un pliego de condiciones iníciales: tenía que ser fácilmente programable y reprogramable, de preferencia en la planta, de fácil mantenimiento y reparación, más pequeño que su equivalente de relevadores, con

un costo que compitiera con los dispositivos de estado sólido y los paneles de relevadores entonces en uso.

Esto provocó un gran interés a los ingenieros de todas las disciplinas inmiscuidos en el control industrial. Un microprocesador basado en PLC se introdujo en 1977 por Allen-Bradley en los EE.UU., usando un microprocesador Intel 8080 con un circuito para manejar las instrucciones lógicas a alta velocidad. Los primeros PLC fueron diseñados sólo para los trabajos de secuenciación basadas en la lógica (señales on/off). Hoy en día hay cientos de diferentes modelos de PLC en el mercado. Se diferencian en su tamaño de memoria que va de 256 bytes a varios kilobytes, y la capacidad de entradas y salidas. El PLC más pequeño puede sustituir a los reléy con temporizador y contador. Muchos PLC's modernos también aceptan señales proporcionales. Se pueden realizar cálculos aritméticos simples, manejar entradas y salidas analógicas además de realizar control PID.

Los PLC actuales pueden comunicarse con otros controladores y computadoras en redes de área local, y son una parte fundamental de los modernos sistemas de control distribuido.

Existen varios lenguajes de programación, tradicionalmente los más utilizados son el diagrama de escalera (Ladder), preferido por los electricistas, lista de instrucciones y programación por estados, aunque se han incorporado lenguajes más intuitivos que permiten implementar algoritmos complejos, mediante simples diagramas de flujo más fáciles de interpretar y mantener. Un lenguaje más reciente, preferido por los informáticos y electrónicos, es el FBD en inglés Function Block Diagram, que emplea compuertas lógicas y bloques con distintas funciones conectados entre sí.

En la programación se pueden incluir diferentes tipos de operandos, desde los más simples como lógica booleana, contadores, temporizadores, contactos, bobinas y operadores matemáticos, hasta operaciones más complejas como manejo de tablas, apuntadores, algoritmos PID y funciones de comunicación multiprotocolo que le permitirían interconectarse con otros dispositivos [13].

2.8 Estructura del PLC

Un controlador lógico programable está constituido por un conjunto de tarjetas o circuitos impresos, sobre los cuales están ubicados componentes electrónicos.

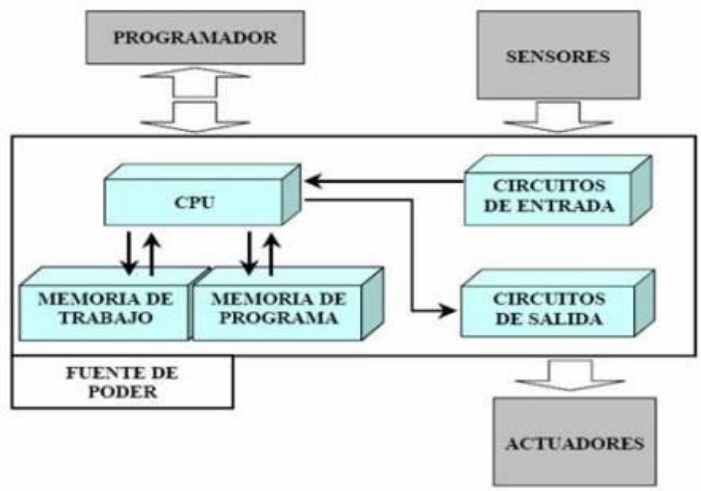


Figura 2.22 Estructura de un PLC [14]

El controlador programable tiene la estructura típica de muchos sistemas programables, como por ejemplo una microcomputadora (figura 2.22). La estructura básica del hardware de un controlador Programable propiamente dicho está constituido por:

- Fuente de alimentación
- Unidad de procesamiento central (CPU)
- Módulos de interfaces de entradas/salidas (E/S)
- Modulo de memorias
- Unidad de programación

En algunos casos cuando el trabajo que debe realizar el controlador es más exigente, se incluyen módulos inteligentes.

2.8.1 Fuente de alimentación

La función de la fuente de alimentación en un controlador, es suministrar la energía al CPU y demás tarjetas según la configuración del PLC.

+ 5 V para alimentar a todas las tarjetas

+ 5.2 V para alimentar al programador

+ 24 V para los canales de lazo de corriente 20 mA.

2.8.2 Unidad de procesamiento central (CPU)

Es la parte más compleja e imprescindible del controlador programable, que en otros términos podría considerarse el cerebro del controlador. La unidad central está diseñada a base de microprocesadores y memorias; contiene una unidad de control, la memoria interna del programador RAM, temporizadores, contadores, memorias internas tipo relé, imágenes del proceso entradas/salidas, etc. Su misión es leer los estados de las señales de las entradas, ejecutar el programa de control y gobernar las salidas, el procesamiento es permanente y a gran velocidad.

2.8.2.1 Unidad de procesamiento central (CPU)

Al comenzar el ciclo, la CPU lee el estado de las entradas. A continuación ejecuta la aplicación empleando el último estado leído. Una vez completado el programa, la CPU ejecuta tareas internas de diagnóstico y comunicación. Al final del ciclo se actualizan las salidas. El tiempo de ciclo depende del tamaño del programa, del número de E/S y de la cantidad de comunicación requerida, ver figura 2.23.

Figura 2.23 Ciclo de trabajo de un PLC

2.8.3 Módulos o interfaces de entrada y salida (e/s)

Son los que proporciona el vínculo entre la CPU del controlador y los dispositivos de campo del sistema. A través de ellos se origina el intercambio de información ya sea para la adquisición de datos o la del mando para el control de máquinas del proceso.

2.8.4 Tipos de módulos de entrada y salida

Debido a que existen gran variedad de dispositivos exteriores (captadores actuadores), encontramos diferentes tipos de módulos de entrada y salidas, cada uno de los cuales sirve para manejar cierto tipo de señal (discreta o análoga) a determinado valor de tensión o de corriente en DC o AC.

Módulos de entradas discretas
Módulos de salidas discretas
Módulos de entrada analógica
Módulos de salida analógica

2.8.5 Módulos de memorias

Son dispositivos destinados a guardar información de manera provisional o permanente. Se cuenta con dos tipos de memorias,

Volátiles (RAM)
No volátiles (EPROM y EEPROM)

2.8.6 Unidad de programación

Los terminales de programación, son el medio de comunicación entre el hombre y la máquina; estos aparatos están constituidos por teclados y dispositivos de visualización. Existen tres tipos de programadores los manuales (Hand Held) tipo de calculadora, Los de video tipo (PC), y la (computadora).

2.8.7 Programación

Los primeros PLC, en la primera mitad de los 80, eran programados usando sistemas de programación propietarios o terminales de programación especializados, que a menudo tenían teclas de funciones dedicadas que representaban los elementos lógicos de los programas de PLC. Los programas eran guardados en cintas. Más recientemente, los programas PLC son escritos en aplicaciones especiales en un ordenador, y luego son descargados directamente mediante un cable o una red al PLC. Los PLC viejos usan una memoria no volátil (magnetic core memory) pero ahora los programas son guardados en una RAM con batería propia o en otros sistemas de memoria no volátil como las memoria flash. Los primeros PLC fueron diseñados para ser usados por electricistas que podían aprender a programar los PLC en el trabajo. Estos PLC eran programados con “lógica de escalera” ("ladder logic"). Los PLC modernos pueden ser programados de muchas formas, desde la lógica de escalera hasta lenguajes de programación tradicionales como el BASIC o C. Otro método es usar la Lógica de Estados (State Logic), un lenguaje de programación de alto nivel diseñado para programas PLC basándose en los diagramas de transición de estados. Mientras que los conceptos fundamentales de la programación del PLC son comunes a todos los fabricantes, las diferencias en el direccionamiento E/S, la organización de la memoria y el conjunto de instrucciones hace que los programas de los PLC nunca se puedan usar entre diversos fabricantes. Incluso dentro de la misma línea de productos de un solo fabricante, diversos modelos pueden no ser directamente compatibles.

2.9 Comunicaciones

Las formas como los PLC intercambian datos con otros dispositivos son muy variadas. Típicamente un PLC puede tener integrado puertos de comunicaciones seriales que pueden cumplir con distintos estándares de acuerdo al fabricante. Estos puertos pueden ser de los siguientes tipos:

- RS-232
- RS-485
- RS-422
- Ethernet

Sobre estos tipos de puertos de hardware las comunicaciones se establecen utilizando algún tipo de protocolo o lenguaje de comunicaciones. En esencia un protocolo de comunicaciones define la manera como los datos son empaquetados para su transmisión y como son codificados. De estos protocolos los más conocidos son:

- Modbus
- Bus CAN
- Profibus
- Devicenet
- Controlnet
- Ethernet I/P

Muchos fabricantes además ofrecen distintas maneras de comunicar sus PLC con el mundo exterior mediante esquemas de hardware y software protegidos por patentes y leyes de derecho de autor.

Capítulo 3
Estado del arte

3 Estado del arte

3.1 Control de automatización programable (PAC)

Un PAC (Programmable Automation Controller) es una tecnología industrial orientada al control automatizado avanzado, al diseño de equipos para laboratorios y a la medición de magnitudes análogas. El PAC se refiere al conjunto formado por un controlador (una CPU típicamente), módulos de entradas y salidas, y uno o múltiples buses de datos que lo interconectan todo. Este controlador combina eficientemente la fiabilidad de control de un autómata o PLC junto a la flexibilidad de monitorización, cálculo y desempeño de un computador industrial.

Los PAC's pueden utilizarse en el ámbito investigador y de laboratorios, pero es sobre todo en el industrial, para control de máquinas y procesos, donde más se utiliza. A destacar los siguientes: múltiples lazos cerrados de control independientes, lazos de control robusto, adquisición de datos de precisión, análisis matemático y memoria profunda, monitorización remota, visión artificial, control de movimiento y robótica, seguridad controlada, administración de recursos ARP o SAP, entre otros [14].

Los PACs de se comunican usando los protocolos de red abiertos como TCP/IP u OPC. Específicamente los PACs Beckhoff prácticamente están abiertos a todos los protocolos industriales como lo son EtherCAT, Lightbus, PROFIBUS DP / FMS, Interbus, CANopen, Multi-Master, DeviceNet, ControlNet, Modbus, Fipio, CC-Link, SERCOS RS232/RS485, Ethernet TCP / IP, Ethernet / IP, PROFINET, USB, entre otros.

Los PACs y PLCs tienen varias cosas en común. Internamente, ambos incluyen una fuente de potencia, un CPU, un plano trasero o dispositivo de E/S, y módulos. Tienen registros de memoria que reflejan los canales de E/S individuales en los módulos. Sin embargo, las siguientes diferencias resultan muy significativas.

En su estudio de “Generalidades de los Controladores Lógicos Programables a Nivel Mundial”, ARC identificó 5 características principales en los PAC:

- Funcionalidad de dominio múltiple, al menos dos de lógica, movimiento, control PID, y proceso en una sola plataforma
- Plataforma de desarrollo sencillo de disciplina múltiple incorporando etiquetas comunes y una base de datos sencilla para tener acceso a todos los parámetros y funciones
- Herramientas de software que permiten diseñar flujo del proceso a través de varias máquinas o unidades de proceso, junto con el IEC 61131-3, guía del usuario y administración de datos
- Arquitecturas modulares, abiertas que reflejan las aplicaciones industriales a partir de un despliegue de maquinaria en fábricas en plantas de proceso
- Uso de estándares de la industria para interfases en red, lenguajes, etc., como búsquedas TCP/IP, OPC y XML, y SQL

Una ventaja de los PAC al compararse con los PLCs, son la habilidad para procesar y desempeñar medidas complejas. Con esta característica, puede combinar diferentes sistemas de adquisición de datos como frecuencias, formas de onda, voltajes, corrientes, control de movimiento e incluso, adquisición de imágenes. Esto crea un nivel sin precedentes de manipulación y estandarización en términos del tipo de señales que pueden manipularse y procesarse. Los PACs ofrece cientos de funciones para procesar, analizar y extraer información de estas señales, además los PACs cuentan con procesadores que les permiten poder ejecutar cientos de iteraciones y cálculos PID simultáneamente, así como controles robustos como redes neuronales o lógica difusa, ver comparativo en tabla VI.

Tabla VI Cuadro comparativo entre PLC, PAC, PC [14]

Características	PLC	PAC	PC Estándar
Soporta shocks eléctricos y vibración	✓	✓	✗
Seguridad y estabilidad	✓	✓	✗
Rangos de temperatura industriales	✓	✓	✗
Trabajo en tiempo real	✓	✓	✗
Entradas de fuente de poder redundantes	✓	✓	✗
Procesador de punto flotante	✗	✓	✓
Memoria no volátil	✗	✓	✓
Conectividad a Ethernet vía WEB	✗	✓	✓
Capacidad de administración de recursos	✗	✓	✓
Capacidad ilimitada de lazos de control	✗	✓	✓

El controlador programable de automatización SNAP-PAC-R1 y Snap-PAC-R2 (figura 3.1) provee control, comunicación, y procesamiento de I/O en un paquete compacto montado en un rack. Uno de cuatro componentes del Sistema SNAP PAC, el SNAP-PAC-R1 se encuentra totalmente integrado al software PAC Project, a los brains SNAP PAC. Utilizado con el software PAC Project Basic, que viene incluido (o el PAC Project Professional, que se obtiene por separado), el SNAP-PAC-R1, basado en topología Ethernet, puede manejar casi por completo todas sus necesidades de control industrial, monitoreo remoto y necesidades de adquisición de dato.

Figura 3.1 Controlador programable de automatización SNAP-PAC-R1

El PAC NI CompactRIO, impulsado por el software NI LabVIEW (figura 3.2), ofrece el rendimiento y la fiabilidad requeridos para aplicaciones de control industrial y embebido. Combina la funcionalidad de una arquitectura abierta y embebida y la fiabilidad de un FPGA de tamaño pequeño, robusto y módulos industriales de E/S intercambiables en vivo. CompactRIO es ideal para diseñar, generar prototipos y desplegar sus soluciones de medidas y control

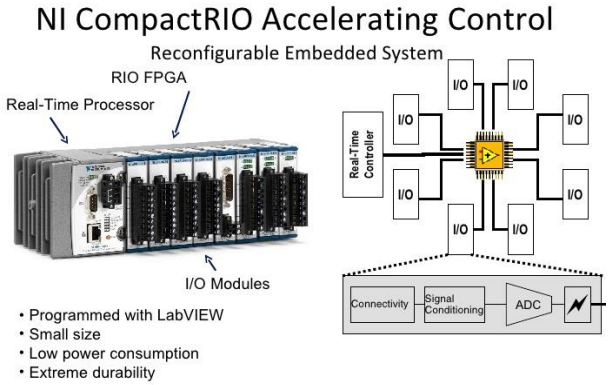


Figura 3.2 PAC NI CompactRIO

3.2 Sistema HMI (Interfaz Hombre-Maquina) y SCADA (Supervisión, Control y Adquisición de Datos)

HMI y SCADA están relacionados entre sí en la medida en que uno o varios HMIs son subconjuntos o componentes de un sistema SCADA. Además, un DCS o Sistema de Control Distribuido es muy similar a un sistema SCADA, y también puede utilizar uno o más HMI s también. Todos estos componentes son clases de, o describen partes de, un ICS o Sistema de Control Industrial, que es la descripción general de la automatización. En los sistemas de control modernos, hay una gran cantidad de tecnología y funcionalidad entre estas dos clases de ICS's.

HMI o interfaz hombre-máquina es simplemente la manera en que los humanos interactúan con las máquinas. Un Sistema SCADA (Supervisory Control

and Data Acquisition) es un sistema de automatización o sistema de control industrial que involucra, control directo o comunicación con uno o más de los siguientes:

- Redes de automatización industrial y máquinas
- Telemetría y control remoto utilizando comunicaciones continuas o ráfaga
- Sistemas de control de procesos y control de procesos estadísticos
- Sistemas de adquisición de datos (DAQ s)
- Históricos y servidores de almacenamiento de datos
- Sistemas de control Industrial utilizando PLCs y RTUs.
- Sistemas del entorno empresarial, tales como sistemas ERP y MES.
- Entorno de computación de nube industrial
- Sistemas de seguridad y procesos
- Seguridad de máquina local
- Seguridad y control de procesos
- Conectividad empresarial o global que implica LDAP y otros.

Un sistema SCADA puede estar conectado continuamente a todos los componentes en el ICS , o puede estar intermitentemente conectado a algunos o todos, y se actualiza con una ráfaga de comunicación a través de modems de radio o celular (tecnologías 2G, 3G o 4G, CDMA y GSM) a los dispositivos y equipos de campo. Un sistema SCADA suele tener uno o más servidores SCADA que contienen una aplicación que está/están comunicando con una ejecución en conjunto con componentes inteligentes tales como PLCs y / o RTU, o posiblemente incluso a distancia, independientemente del sistema SCADA. En la figura 3.3 se muestra un ejemplo de un sistema HMI SCADA implementado.

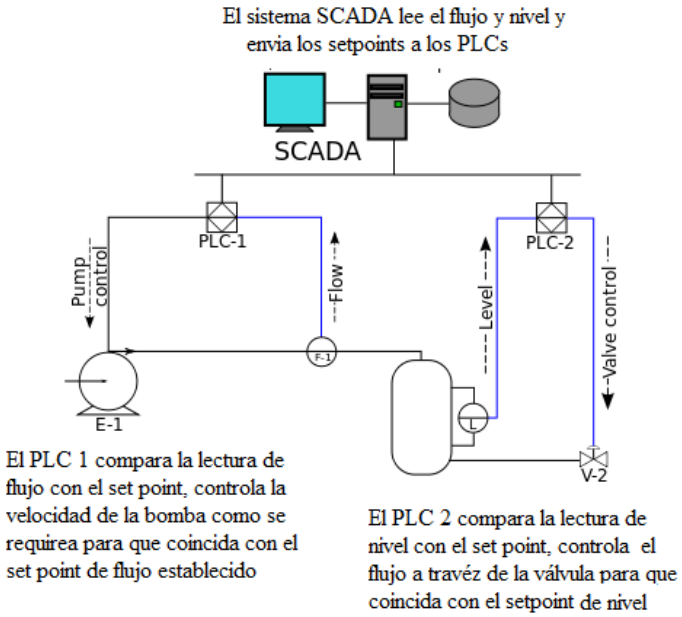


Figura 3.3 Ejemplo de un sistema HMI [14]

3.3 Microcontroladores Arduino

Arduino es una plataforma de prototipos electrónica de código abierto (open-source) basada en hardware y software flexibles y fáciles de usar. Arduino puede sentir el entorno mediante la recepción de entradas desde una variedad de sensores y puede afectar a su alrededor mediante el control de luces, motores y otros artefactos. El microcontrolador de la placa se programa usando el Arduino Programming Language (basado en Wiring) y el Arduino Development Environment (basado en Processing). Los proyectos de Arduino pueden ser autónomos o se pueden comunicar con software en ejecución en un ordenador (por ejemplo con Flash, Processing, MaxMSP, etc.).

Las placas se pueden ensamblar a mano o encargarlas pre ensambladas; el software se puede descargar gratuitamente. Los diseños de referencia del hardware

(archivos CAD) están disponibles bajo licencia open-source, por lo que eres libre de adaptarlas a tus necesidades.

Las principales ventajas de estas placas y otros tipos de micro controladores son las siguientes:

- Las placas Arduino son relativamente baratas comparadas con otras plataformas de Microcontroladores.
- Se ejecuta en sistemas operativos Windows, Macintosh OSX y GNU/Linux. La mayoría de los sistemas de Microcontroladores están limitados a Windows.
- El entorno de programación de Arduino es fácil de usar para principiantes, pero suficientemente flexible para que usuarios avanzados puedan aprovecharlo también. Para profesores, está convenientemente basado en el entorno de programación Processing, de manera que estudiantes aprendiendo a programar en ese entorno estarán familiarizados con el aspecto y la imagen de Arduino.
- El software Arduino está publicado como herramientas de código abierto, disponible para extensión por programadores experimentados. El lenguaje puede ser expandido mediante librerias C++, y la gente que quiera entender los detalles técnicos pueden hacer el salto desde Arduino a la programación en lenguaje AVR C en el cual está basado. De forma similar, puedes añadir código AVR-C directamente en tus programas Arduino si quieres.
- El Arduino está basado en Microcontroladores ATMEGA8 y ATMEGA168 de Atmel. Los planos para los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores experimentados de circuitos pueden hacer su propia versión del módulo, extendiéndolo y mejorándolo. Incluso usuarios relativamente inexpertos pueden construir la versión de la placa del módulo para entender cómo funciona y ahorrar dinero.

3.3.1 Arduino UNO

"El Arduino Uno R3 utiliza el microcontrolador ATmega328. En adición a todas las características de las tarjetas anteriores, el Arduino Uno utiliza el ATmega16U2 para el manejo de USB en lugar del 8U2 (o del FTDI encontrado en generaciones previas). Esto permite ratios de transferencia más rápidos y más memoria. No se necesitan drivers para Linux o Mac (el archivo inf para Windows es necesario y está incluido en el IDE de Arduino).

La tarjeta Arduino Uno R3 incluso añade pins SDA y SCL cercanos al AREF. Existen dos nuevos pines cerca del pin RESET. Uno es el IOREF, que permite a los shields adaptarse al voltaje brindado por la tarjeta. El otro pin no se encuentra conectado y está reservado para propósitos futuros. La tarjeta trabaja con todos los shields existentes y podrá adaptarse con los nuevos shields utilizando esos pines adicionales. El Arduino es una plataforma computacional física open-source basada en una simple tarjeta de I/O y un entorno de desarrollo que implementa el lenguaje Processing/Wiring. El Arduino Uno R3 puede ser utilizado para desarrollar objetos interactivos o puede ser conectado a software de tu computadora (por ejemplo, Flash, Processing, MaxMSP). El IDE open-source puede ser descargado gratuitamente (actualmente para Mac OS X, Windows y Linux).

Características:

- Microcontrolador ATmega328.
- Voltaje de entrada 7-12V.
- 14 pines digitales de I/O (6 salidas PWM).
- 6 entradas análogas.
- 32k de memoria Flash.
- Reloj de 16MHz de velocidad."

En la figura 3.4 se muestra un Arduino UNO.

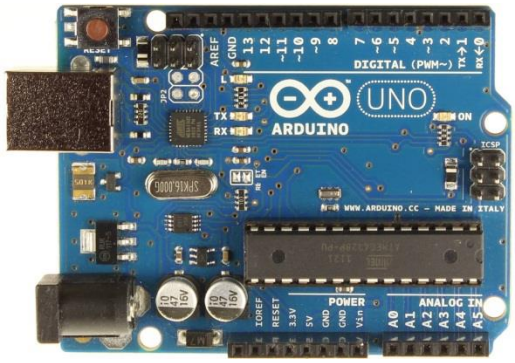


Figura 3.4 Arduino UNO

3.3.2 Arduino MEGA

"El Arduino Mega 2560 es una placa basado en el ATmega2560. Cuenta con 54 pines digitales de entrada / salida (de los cuales 15 se pueden utilizar como salidas PWM), 16 entradas analógicas, 4 UART (puertos serie de hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP, Y un botón de reinicio. Contiene todo lo necesario para soportar el microcontrolador; Simplemente conéctelo a un ordenador con un cable USB o conéctelo con un adaptador de CA a CC o batería para empezar. La placa Mega 2560 es compatible con la mayoría de los shields diseñados para el Uno y las placas anteriores Duemilanove o Diecimila.

Características:

- Microcontrolador ATmega1280
- Voltaje de operación 5 V
- Voltaje de entrada (recomendado) 7-12 V
- Voltaje de entrada (limites) 6-20 V
- Pines Digitales I/O 54 (15 provén salida PWM)
- Pines Analógicos de entrada 16
- Corriente DC por Pin I/O 40 mA
- Corriente DC para Pin 3.3V 50 mA
- Memoria Flash 128 KB

- SRAM 8 KB
- EEPROM 4 KB
- Velocidad de reloj 16 MHz”

En la figura 3.5 se muestra el Arduino Mega, (Ver anexo 11).

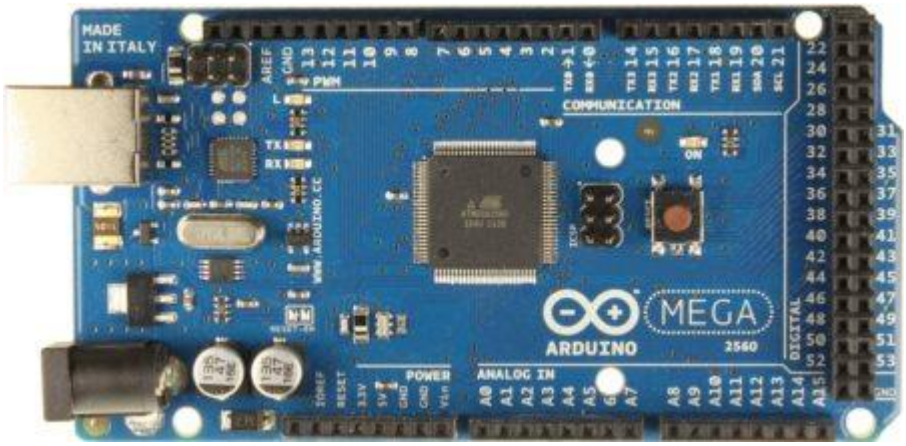


Figura 3.5 Arduino MEGA

3.4 Sensor de temperatura

Los sensores de temperatura son dispositivos que transforman los cambios de temperatura en señales eléctricas que son procesados por equipo eléctrico o electrónico. Hay tres tipos de sensores de temperatura, los termistores, los RTD y los termopares.

El sensor de temperatura, típicamente suele estar formado por el elemento sensor, de cualquiera de los tipos anteriores, la vaina que lo envuelve y que está rellena de un material muy conductor de la temperatura, para que los cambios se transmitan rápidamente al elemento sensor y del cable al que se conectarán el equipo electrónico, a continuación se muestran dos sensores digitales.

3.4.1 Sensor Ds18b20

El sensor de temperatura DS18B20 (ver figura 3.6), es un dispositivo que se comunica de forma digital. Cuenta con tres terminales: Vcc, GND y el pin Data. Este sensor utiliza comunicación por protocolo serial digital OneWire. Esté protocolo de comunicación permite enviar y recibir datos utilizando un solo cable. A diferencia de otros, que utilizan dos o más líneas de comunicación digital. Para leer el sensor con un Arduino es necesario utilizar dos librerías que deben ser instaladas antes de cargar el código a nuestra placa de desarrollo. Entonces las librerías son las siguientes:

- Dallas Temperature.
- OneWire

Sus características son:

- Sensor Digital.
- Resolución de 9 y 12 bits.
- Rango de operación de -50 a 125 grados Centígrados.
- Precisión +- 0.5 grados.
- Protocolo OneWire.

Figura 3.6 Sensor Ds18b20

3.4.2 Sensor DHT22

Los sensores DHT22 (figura 3.7), son sensores digitales de temperatura y humedad, fáciles de implementar con cualquier microcontrolador. Utiliza un sensor capacitivo de humedad y un termistor para medir el aire circundante y solo un pin para la lectura de los datos. La desventaja de estos es la velocidad de las lecturas y el tiempo que hay que esperar para tomar nuevas lecturas (cada 2 segundos), pero esto no es tan importante puesto que la temperatura y humedad son variables que no cambian muy rápido en el tiempo. El coeficiente de calibración se guarda en el tipo de programa en la memoria OTP. Cuando el sensor detecta, citará coeficiente de memoria.

Con un tamaño pequeño, bajo consumo y larga distancia de transmisión (20m) permiten que DHT22 sea adecuado para todo tipo de ocasiones difíciles de aplicación. Una sola fila empaquetada con cuatro pines, hace que la conexión sea muy conveniente.

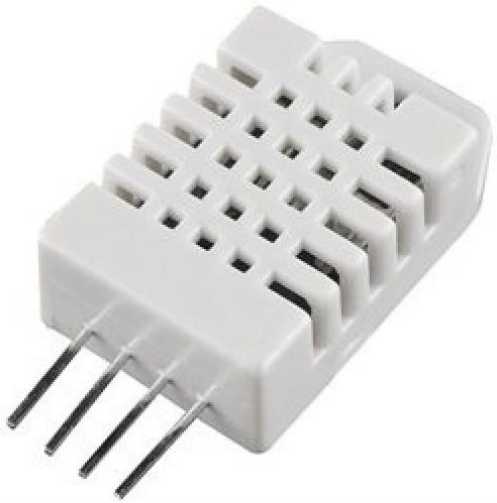


Figura 3.7 Sensor DTH 22

3.5 Software LabVIEW

LabVIEW es un lenguaje completamente gráfico, y el resultado de ello es que es totalmente parecido a un instrumento, por ello a todos los módulos creados con LabVIEW se les llama VI (Instrumento Virtual). Existen dos conceptos básicos en LabVIEW: el Front Panel (Panel Frontal) y el Block diagram (Diagrama de Bloque).

El panel frontal es el interfaz que el usuario está viendo y puede ser totalmente parecido al instrumento del cual se están recogiendo los datos, de esta manera el usuario sabe de manera precisa cual es el estado actual de dicho instrumento y los valores de las señales que se están midiendo.

El diagrama de bloques es el conexionado de todos los controles y variables, que tendría cierto parecido al diagrama del esquema eléctrico del instrumento. LabVIEW tiene la característica de descomposición modular ya que cualquier VI que se ha diseñado puede convertirse fácilmente en un módulo que puede ser usado como una sub-unidad dentro de otro VI.

Esta peculiaridad podría compararse a la característica de procedimiento en los lenguajes de programación estructurada. Es un sistema abierto, en cuanto a que cualquier fabricante de tarjetas de adquisición de datos o instrumentos en general puede proporcionar el driver de su producto en forma de VI dentro del entorno de LabVIEW. También es posible programar módulos para LabVIEW en lenguajes como C y C++, estos módulos son conocidos como Sub-VIs y no se difieren a los VI creados con LabVIEW salvo por el interfaz del lenguaje en el que han sido programados. Además estos Sub-VIs son muy útiles por ejemplo en el campo de cálculos numéricos complejos que no se encuentran incluidos en las librerías de LabVIEW.

3.6 Software Matlab

MATLAB (abreviatura de MATrix LABoratory, "laboratorio de matrices") es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux .

Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes y con otros dispositivos hardware. El paquete MATLAB dispone de dos herramientas adicionales que expanden sus prestaciones, a saber, Simulink (plataforma de simulación multidominio) y GUIDE (editor de interfaces de usuario - GUI). Además, se pueden ampliar las capacidades de MATLAB con las cajas de herramientas (toolboxes); y las de Simulink con los paquetes de bloques (blocksets).

Es un software muy usado en universidades y centros de investigación y desarrollo. En los últimos años ha aumentado el número de prestaciones, como la de programar directamente procesadores digitales de señal o crear código VHDL.

En 2004, se estimaba que MATLAB era empleado por más de un millón de personas en ámbitos académicos y empresariales

Las aplicaciones de MATLAB se desarrollan en un lenguaje de programación propio. Este lenguaje es interpretado, y puede ejecutarse tanto en el entorno interactivo, como a través de un archivo de script (archivos *.m). Este lenguaje permite operaciones de vectores y matrices, funciones, cálculo lambda, y programación orientada a objetos.

3.7 Nebulizadores para invernadero

Los nebulizadores producen niebla fina, el agua a presión sale por un orificio de pequeño diámetro, de forma que el chorro producido se estrella contra una pared cóncava que lo despide y distribuye en forma nebulizada. Estos sistemas suelen trabajar con presiones relativamente elevadas, en torno a 2-4 bares.

Su uso puede realizarse para múltiples aplicaciones:

- Aumentar la humedad relativa de un invernadero.
- Para refrigerar el invernadero combinado con un sistema de ventilación forzada.
- Para aplicar tratamientos automatizados como la aplicación de abonos foliares, fitosanitarios, o cualquier otro producto soluble en agua.
- Efectuar el riego por nebulización

Estos sistemas pueden ajustar los caudales y el tamaño de gota cambiando la boquilla, para realizar un uso u otro según las necesidades de la producción. Regulando las presiones de agua en el cabezal también se consigue el mismo objetivo, adaptándose la aplicación a la realización humidificación, refrigeración, riego y/o aplicación de abonos o fitosanitarios. Se utiliza, principalmente, en el riego de semilleros e invernaderos.

3.8 Humificadores para invernadero

Son aparatos especiales para usarlos en invernaderos aumentando la humedad en el ambiente. Existen distintos tipos de estos humidificadores, algunos son los centrífugos, de rocío, ultrasónicos, entre muchos otros. Gracias a estos humidificadores potentes y prácticos se logra mejorar la calidad de los productos, y otros beneficios como menos residuos, supresión de polvo y velocidades de producción más rápido. En la figura 3.8 se muestra un humificador AirWet como el empleado en este proyecto.

Figura 3.8 Humificador AirWet

Capítulo 4
Materiales y métodos

4 Materiales y métodos

4.1 Arquitectura de control

La arquitectura de control de la red de sensores de temperatura ds18b20 se presenta en la figura 4.1.

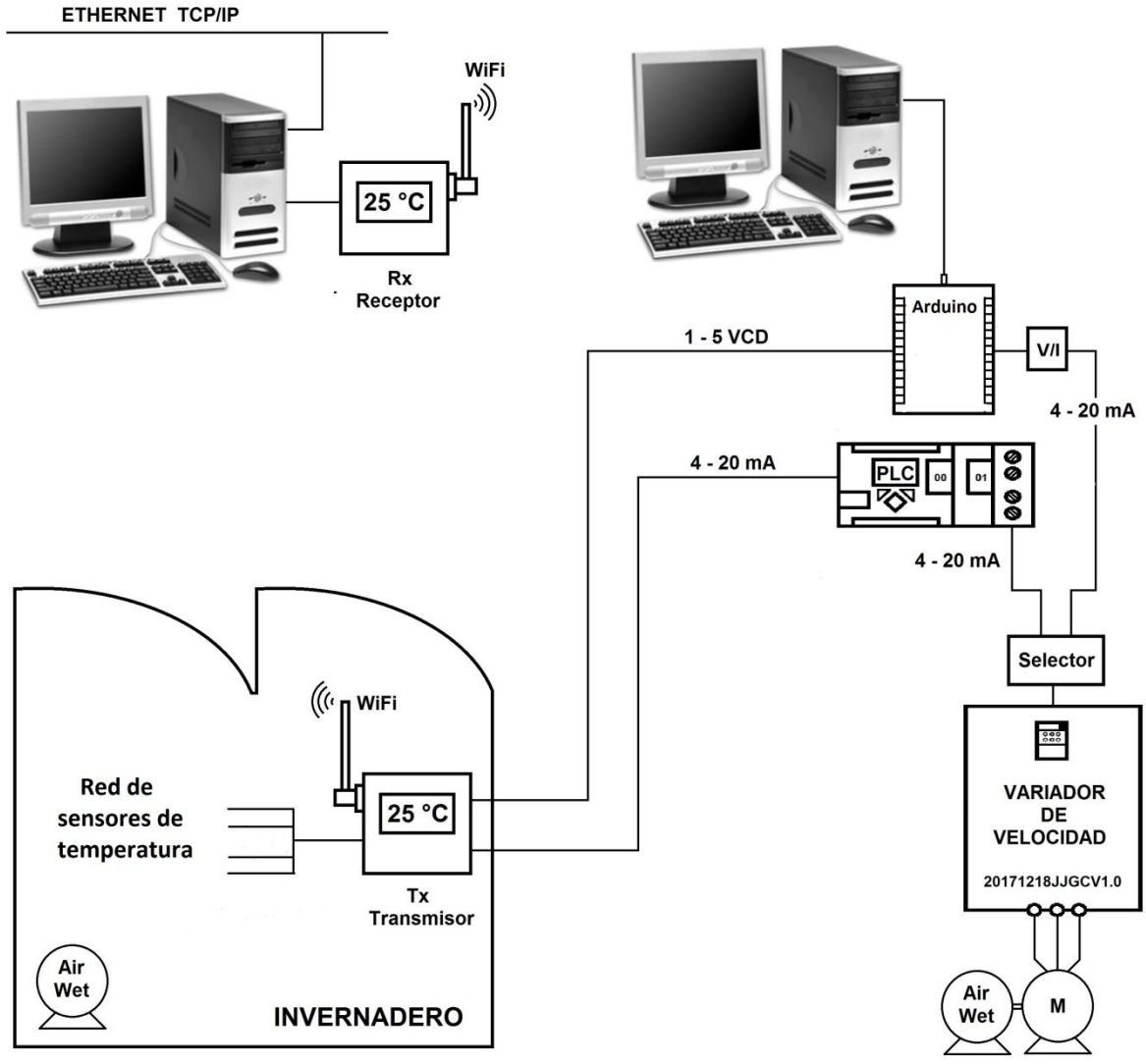


Figura 4.1 Arquitectura de control de red de sensores ds18b20.

4.2 Humificador Airwet

Las características del humificador son las siguientes:

- Ventilador axial de caudal variable
- Nebulización por sistema centrífugo de altas revoluciones
- Ventilación independiente del sistema centrífugo
- Micronización VMD 20 micras
- Presión necesaria de 2 a 6 atm
- Consumo regulable de 10 a 44 l/ha
- Cobertura de humidificación 275/400 m^3 por unidad
- Tratamiento 250/350 m^3 por unidad
- Desinfección 350/450 m^3 por unidad
- Tobera 615x440. Aspiración aerodinámica con radio 50 mm.
- Hélice de paso regulable. Bajo nivel sonoro.
- Motor 0,50 hp, 50 Hz IP65, 1350 rpm. Monofásico o trifásico.
- Sistema centrífugo a 3.000 rpm mediante discos cóncavos inox.
- Turbo inyector Ø5mm interior, sin posibilidad de obstrucciones.
- Válvula reguladora de caudal.
- Rejas protectoras ISO EN.
- Colector de sobrante.
- Bomba recuperadora.

En la figura 4.2 se muestra un Airwet.

Figura 4.2 Airwet

4.3 Sensor Ds18b20

El sensor de temperatura DS18B20 (figura 4.3) es un dispositivo que se comunica de forma digital. Cuenta con tres terminales: Vcc, GND y el pin Data. Este sensor utiliza comunicación por protocolo serial digital OneWire. Esté protocolo de comunicación permite enviar y recibir datos utilizando un solo cable. A diferencia de otros, que utilizan dos o más líneas de comunicación digital. Para leer el sensor con un Arduino es necesario utilizar dos librerías que deben ser instaladas antes de cargar el código a nuestra placa de desarrollo. Entonces las librerías son las siguientes:

- Dallas Temperature.
- OneWire

Sus características son:

- Sensor Digital.
- Resolución de 9 y 12 bits.
- Rango de operación de -50 a 125 grados Centígrados.
- Precisión +- 0.5 grados.
- Protocolo OneWire.

Figura 4.3 Sensor de temperatura Ds18b20

El diagrama de conexión que se emplea es en Bus, debido a que es un grupo de sensores conectados en serie, este diagrama se presenta en la figura 4.4.

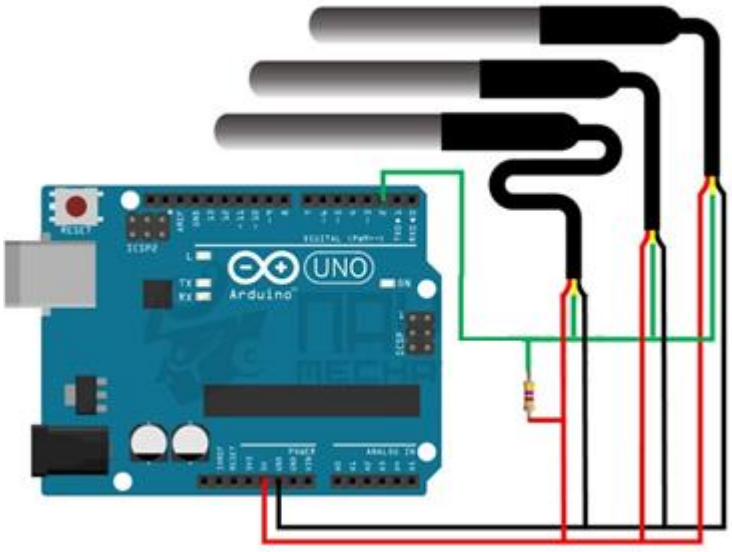


Figura 4.4 Conexión de varios sensores a un pin de Arduino

4.4 Variador de frecuencia General Electric: Micro Drives AF-60

El AF-60 LP (figura 4.5) está construido para todas aquellas aplicaciones de frecuencia variable que requieren el micro rendimiento de un micro drive. El AF-60 LP hace que la instalación y la configuración sean lo más sencillo del mundo. Cuenta con un kit opcional de montaje en riel DIN disponible hasta 3HP.

Figura 4.5 Variador AF-60

Características:

- 230VCA, Monofásico: 1/4HP-3HP
- 230VCA, Trifásico: 1/3HP-5HP
- 230VCA, Trifásico: 1/2HP-30HP
- Características de auto-protección
- 150% de protección para sobrecargas hasta por 1 minuto
- Inicio rápido (atrapa un motor que gira)
- Función de detención precisa
- Sobrecarga térmica electrónica
- Software de PC fácil de usar

- Potenciómetros en el teclado de control
- Envolvente robusto (IP20) que protege el drive y permite el montaje de varios variadores sin existir espacio entre ellos
- Tarjetas de circuitos con un recubrimiento epóxico y capacitores de alta calidad para maximizar el tiempo de funcionamiento
- El manejo inteligente del calor contribuye a una vida útil más prolongada
- Cumple con la norma RoHS para que se considere un producto con responsabilidad ambiental
- Normas: CE, UL, cUL

4.5 Creación de red neuronal en Matlab

La red neuronal para el control del variador se desarrolló en Matlab, utilizando la herramienta “nntool”, para la cual se necesitó la introducción del vector objetivo y el vector de entradas. El vector de entrada se propuso un vector de 1x55, en donde se estimaban las posibles entradas que la neurona pudiese recibir, para el vector de objetivos se utilizó la interpolación lineal para calcular la salida deseada para cada entrada:

$$Y = \left(\frac{(X - X_1)(Y_2 - Y_1)}{(X_2 - X_1)}\right) + Y_1 \ , \qquad (2-1)$$

donde:

- *X1,Y1* = Primeras coordenadas,
- *X2,Y2* = Segundo coordenadas,
- *X* = Coordenada del objetivo X,
- *Y* = Coordenada interpolada.

El procedimiento para la creación y entrenamiento da la red se ejemplifica a continuación:

En la ventana de comando se tecla “nntool”. En la ventana que aparece (figura 4.6) se importan los vectores de entrada y objetivo, posteriormente se clickea en el botón *new*.

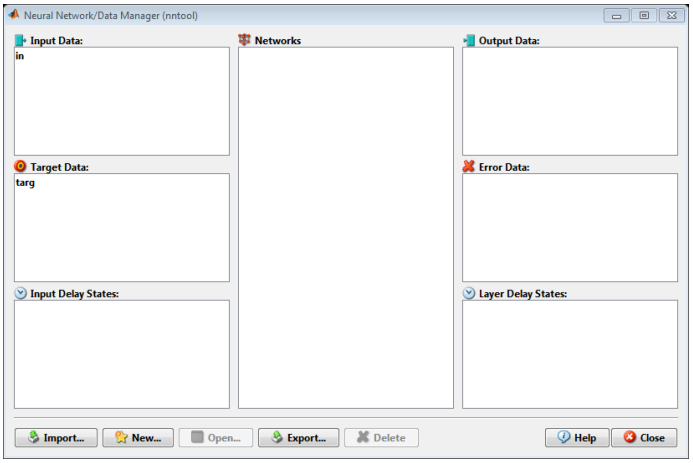


Figura 4.6 Ventana nntool

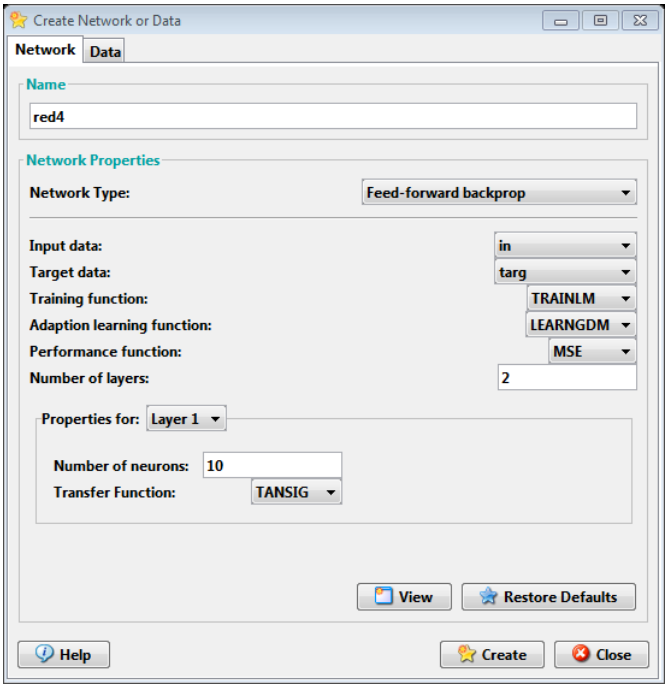


Figura 4.7 Ventana de creación de red neuronal

En la ventana emergente (figura 4.7) se configura el nombre y tipo de la red neuronal, así como el número de neuronas, capas, función de entrenamiento, función de aprendizaje y función de rendimiento. Una vez creada la red se da doble click sobre ella para comenzar con el entrenamiento en las figuras 4.8, 4.9 y 4.10 se muestran los pasos para configurar y entrenar la red:

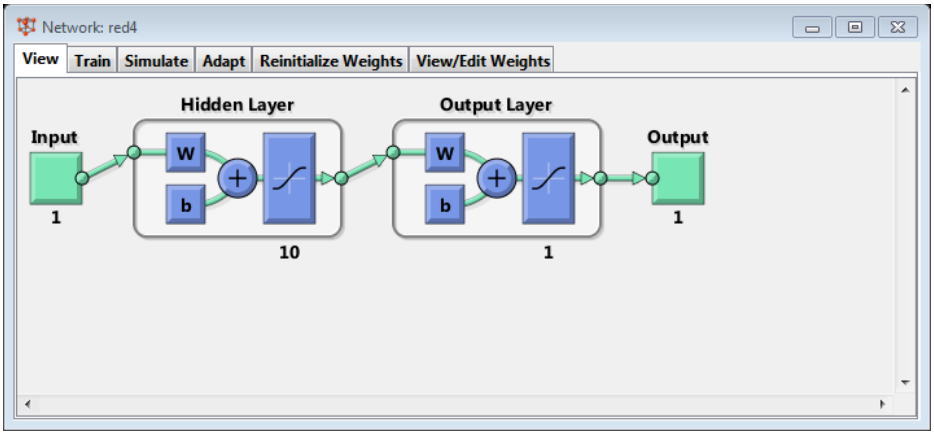


Figura 4.8 Esquema de la red neuronal

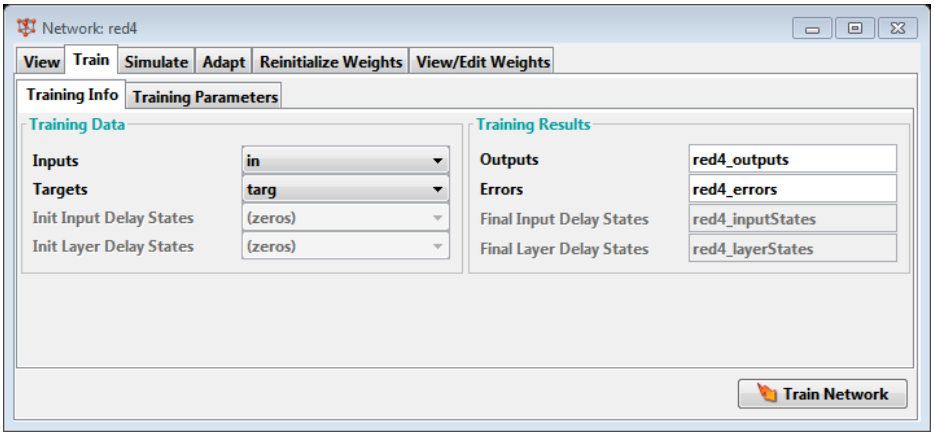


Figura 4.9 Configuración de entradas y objetivos de la red

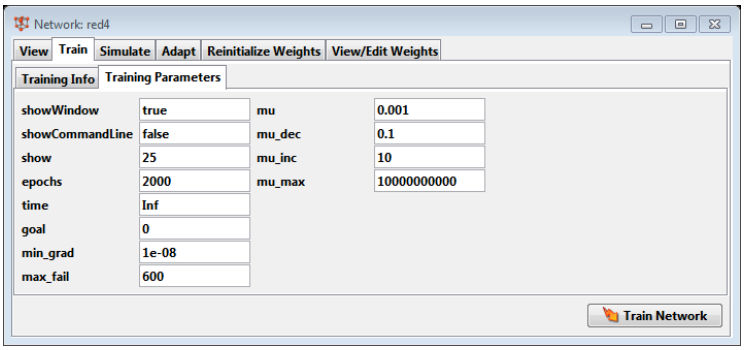


Figura 4.10 Configuración de los parámetros de entrenamiento

La neurona debe entrenarse hasta que se obtenga el rendimiento deseado, es decir se debe entrenar múltiples ocasiones para obtener un resultado aproximado a los del vector objetivo.

El vector de entradas estará formado por los posibles valores que pueda tomar la entrada a la planta, en este caso se tomó el valor del error. Siendo este la diferencia entre el valor de temperatura medido y el Set Point requerido. Por consecuencia el valor del vector objetivo deberá ser los valores deseados para cada valor de entrada, configurando un entrenamiento que permita la adaptabilidad a diferentes situaciones posibles.

4.6 Caracterización de Planta para control PID del invernadero

Para obtener la ecuación de la planta del invernadero se realizó un muestreo de la temperatura en un periodo de tiempo determinado, la prueba consistió en encender el Airwet cuando la temperatura del invernadero se encontraba en su punto más alto, posteriormente se obtuvieron diferentes lecturas de temperatura hasta que se consideró que esta se asentaba. La figura 4.11 muestra el grafico de estas lecturas.

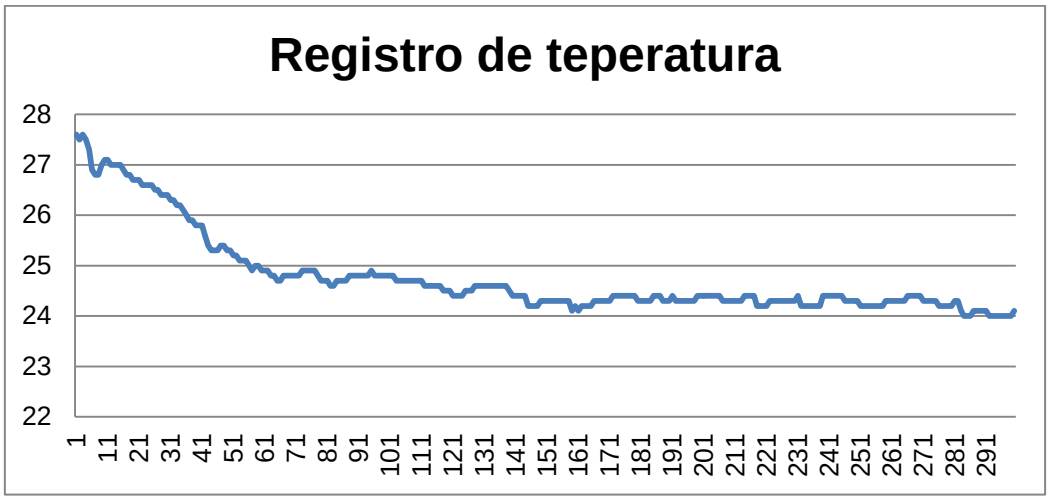


Figura 4.11 Lecturas de temperatura en el invernadero para caracterizar la planta

Para obtener la ecuación de la planta (G) se utilizó el software Matlab con su comando "*ident"*, este comando permite obtener una ecuación de transferencia de una planta a partir de un gráfico de la respuesta de la variable.

El primer paso para obtener la ecuación de transferencia es teclear en la ventana de comandos el comando "*ident*". Lo que abrirá una ventana como la mostrada en la figura 4.12

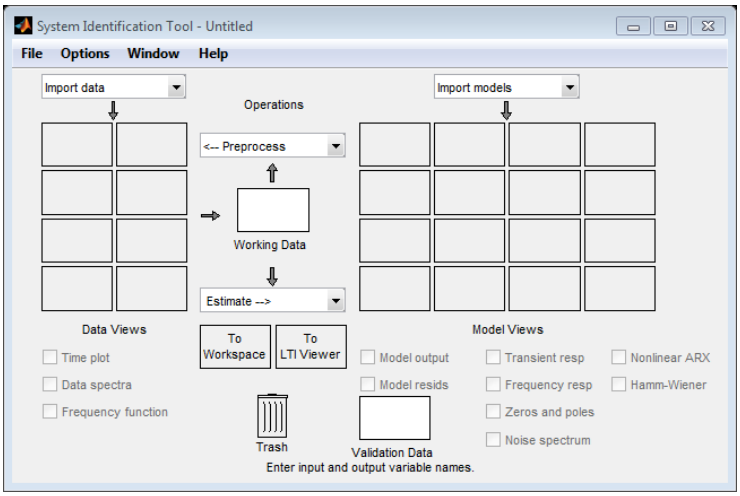


Figura 4.12 Ventana principal del comando ident

En seguida seleccionamos el combo box de “import data” la opción de “time domain data”, debido a que nuestras lecturas fueron tomadas en el domino del tiempo, ver figura 4.13. y posteriormente aparecerá un cuadro en donde se configurara el tiempo de muestreo y el vector de entrada y salida que se adquirieron (figura 4.14).

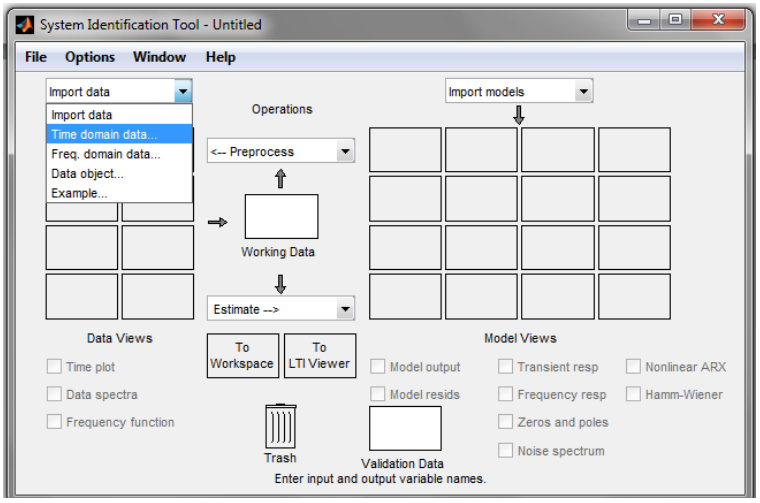


Figura 4.13 Selección de tipo de grafica como datos de entrada

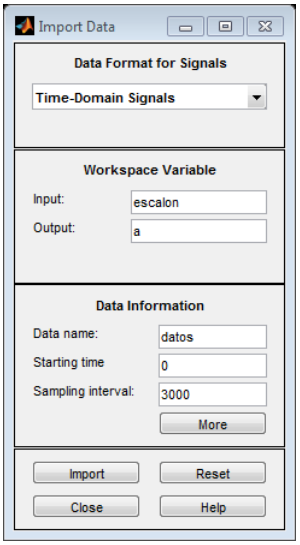


Figura 4.14 Configuración de datos para obtención de planta G

Los pasos siguientes después de haber cargado los datos del workspace, es verificar que los vectores que se hayan ingresado sean los correspondientes a los obtenidos en campo, para ello se selecciona la casilla “time plot”, y su grafico se muestra en la figura 4.15. Como se observa nuestra entrada será un escalón y la salida es la temperatura muestreada en el invernadero. Para obtener la aproximación de la planta se selecciona la opción “Process Models” del combo box de la pantalla principal, como se muestra en la figura 4.16.

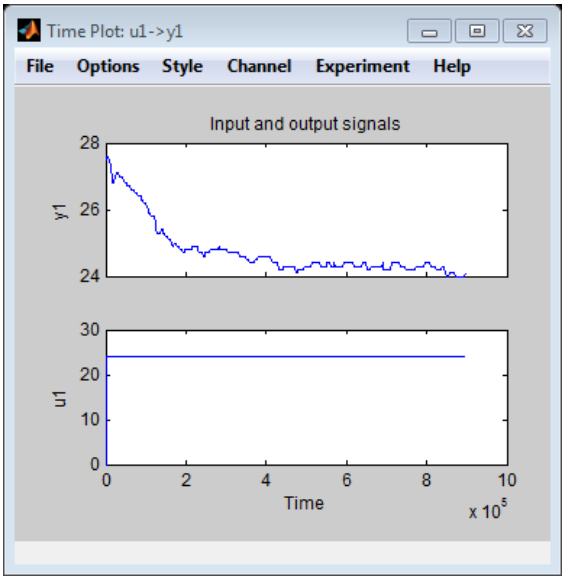


Figura 4.15 Grafica de temperatura obtenida en invernadero

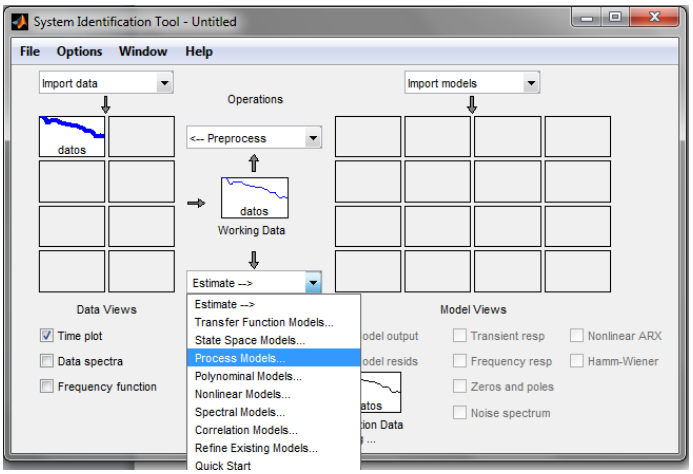


Figura 4.16 Selección de opción Process Models

La ventana que aparecerá en nos indicara la forma estimada de la función de transferencia según los datos ingresados (figura 4.17), este proceso fue repetido hasta obtener una función que tuviera un índice de fidelidad o aproximación mayor al 75% como se muestra en la figura 4.18.

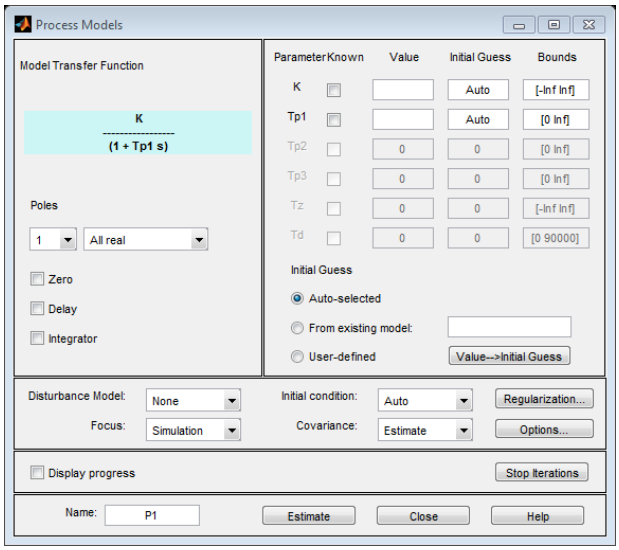


Figura 4.17 Configuración de valores de planta del invernadero

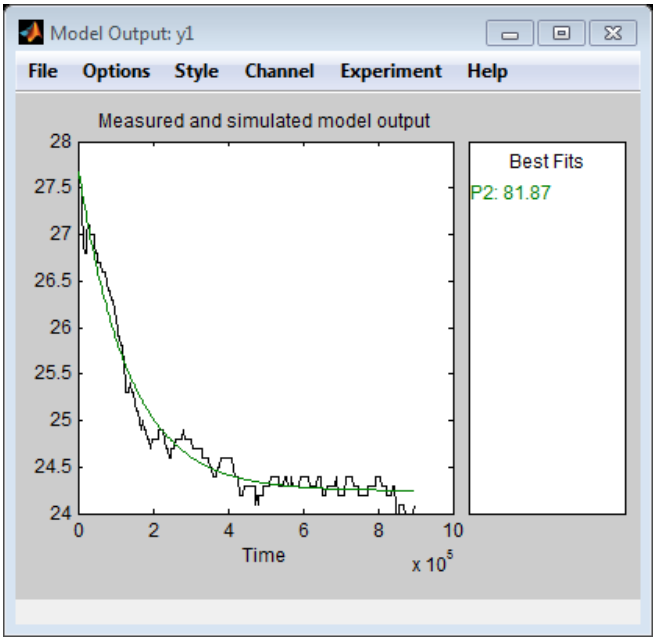


Figura 4.18 Grafico del modelo de salida con un índice de aproximación de 81%

Por consecuente se obtuvo una planta con estas características:

$$(4-2)$$

$$G(s) = \frac{Kp}{1 + Tps * s} ,$$

Donde:

- *Kp*= 1.0098;
- *Tps*=45.8;

$$(4-3)$$

$$G(s) = \frac{1.009}{1 + 45.8s}; ,$$

4.7 Creación de HMI en LabVIEW para control neuronal y PID

La interface humano-maquina o HMI por sus siglas en ingles fue desarrollado por medio del software LabVIEW. El propósito de esta interfaz es proporcionar al usuario una forma cómoda para visualizar los datos adquiridos, así como manipular parámetros para su control remoto, esta interfaz está asociada con el receptor a su vez que emite la señal de control hacia el emisor, es decir la comunicación será bilateral. En las figuras 4.19 y 4.20 se muestran las ventanas de las interfaces a las cuales el usuario tendrá acceso.

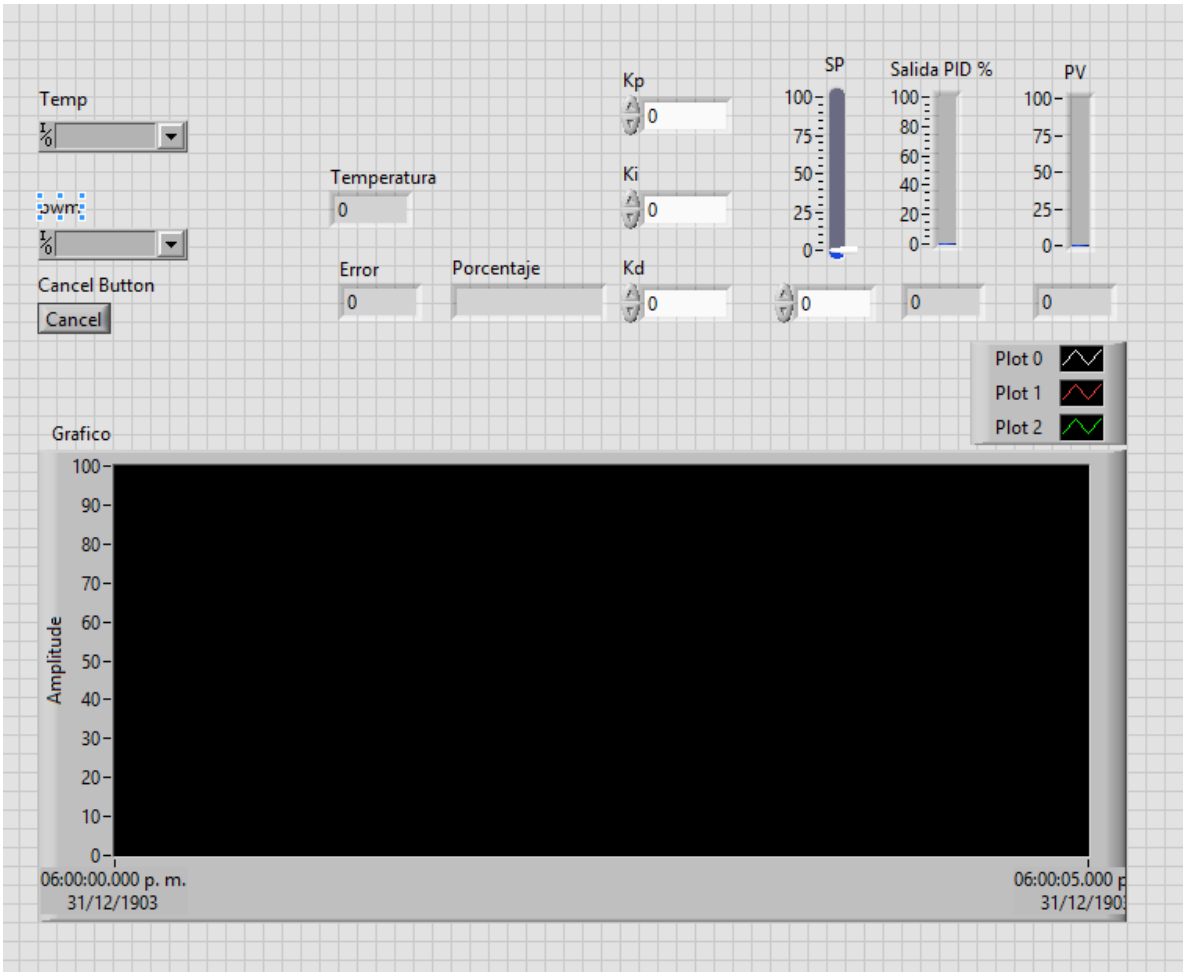


Figura 4.19 Ventana de interfaz para control PID

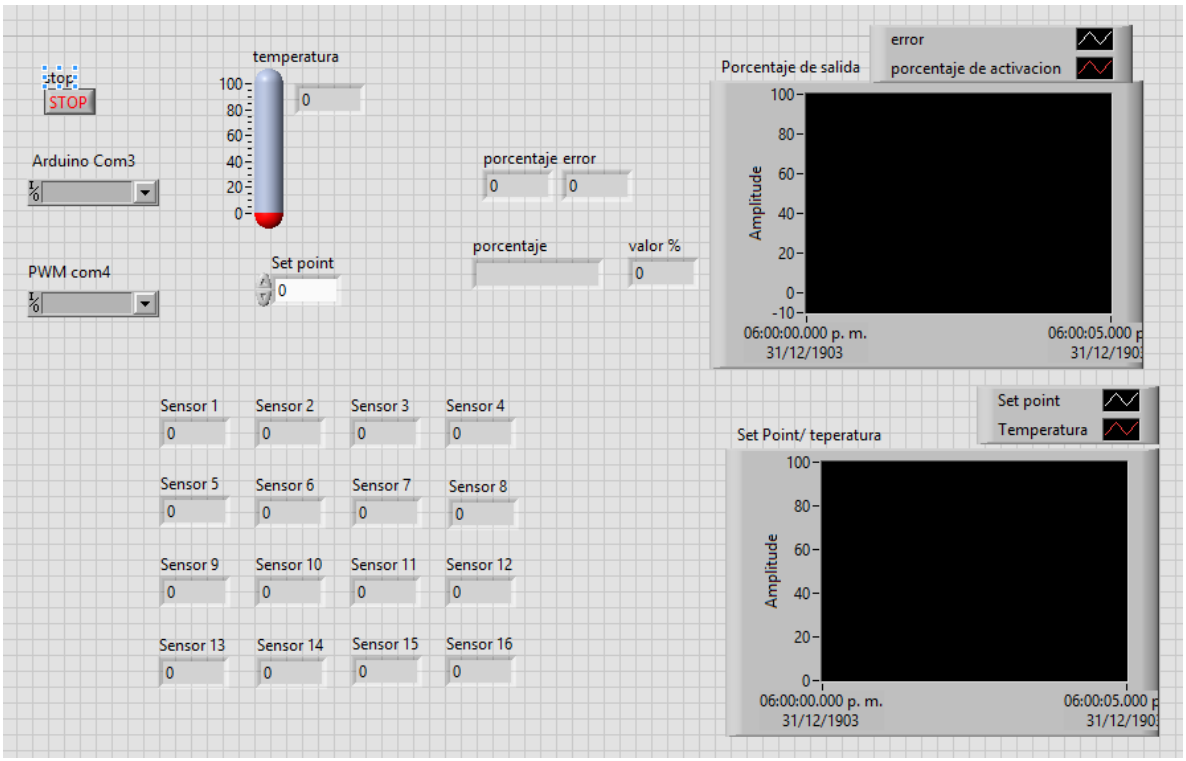


Figura 4.20 Ventana de interfaz control neuronal

Cada interfaz cuenta con un bloque de control distinto, es decir la interfaz del control con la red neuronal cuenta con un bloque de enlace a Matlab para cargar la red, y poder hacer el control, por otro lado la interfaz de control PID tiene un bloque de control previamente configurado para poder realizar su labor, los valores de los parámetros *Kp*, *Ki* y *Kd* pueden ser configurados desde la ventana principal de la interfaz.

4.8 Control neuronal propuesto

La adquisición de datos para realizar el control se lleva a cabo por medio de dieciséis sensores de temperatura Ds18b20 dispuestos en columnas de cuatro dentro del invernadero del Instituto Tecnológico de Ciudad Guzmán (ver figura 4.21), una vez tomadas las lecturas el modulo emisor conformado por un Arduino uno y una antena nrf24l01 realiza un promedio, este valor será el utilizado para establecer los parámetros de control, una vez obtenido la temperatura promedio dentro del invernadero se envía junto con cada lectura individual a un módulo receptor por medio de radio frecuencia , este también compuesto por los mismos componentes que el transmisor, una vez hecha la recepción los datos son enviados a la interfaz hecha en LabVIEW, en donde se configura el parámetro de Set Point, la diferencia entre la variable de temperatura y el Set Point nos dará un valor llamado error, este valor es el ingresado a la red neuronal.

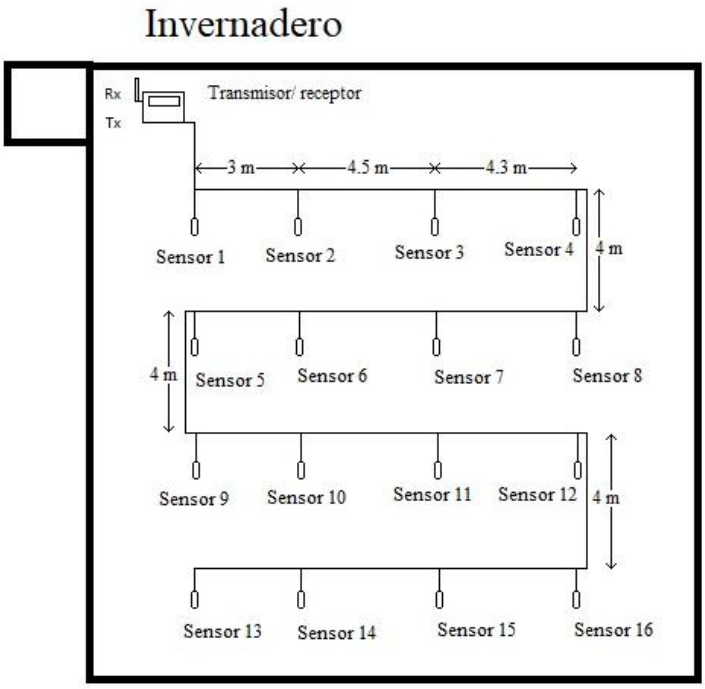


Figura 4.21 Diagrama propuesto para red de sensores

Para el manejo de la red neuronal el error corresponde a un valor del vector entrada para nuestra red, en la figura 4.22 se muestra un esquema de una red neuronal.

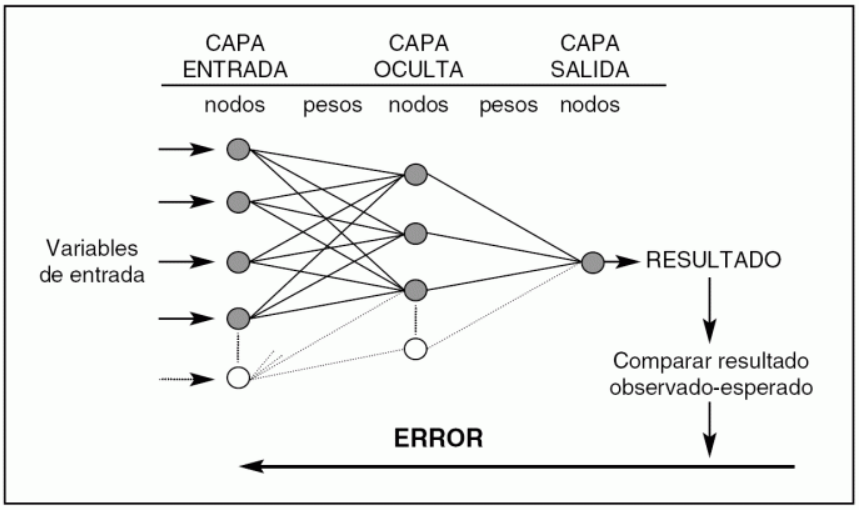


Figura 4.22 Diagrama de Red Neuronal

El vector de entrada propuesto fue de 55 valores, desde el -10 hasta el 100, indicando las posibles cantidades que el error podía adoptar, para el vector objetivo o salidas deseadas se estableció un rango de 0 a 100%, lo que permite enviar una señal de control al variador de velocidad y este a su vez controlar la velocidad del motor del Airwet, el valor de los pesos es asignado por el propio Matlab. La red neuronal utilizada es del tipo feedforward backprop, teniendo 10 neuronas en la capa oculta y un en la capa de salida. El motivo por el cual se le enseño a la red a valores negativos del error fue para asegurar que el motor no trabajaría si el error es negativo o cero, mencionado que el error en este proceso está dado por la ecuación:

$$(4-4)$$

$$error = Vp - Sp$$

Dónde:

Vp= variable de proceso = temperatura

Sp= Set Point = valor de temperatura deseado

Dado que es un control de temperatura cuyo único actuador son elementos para disminuirla, siempre vamos a requerir que el error sea mayor a cero. Para evitar que el motor sufra algún daño debido a baja frecuencia de trabajo, se limitó a que el porcentaje de trabajo mínimo de 36% que equivale a frecuencia de 23.9 Hz. (ver tabla II).

Una vez realizado el cálculo mediante la red neuronal, el dato es enviado de nuevo por vía Serial a un Arduino uno que sirve como emisor, en esta parte se calcula el ancho de pulso correspondiente al valor de salida obtenido, este dato es enviado mediante radiofrecuencia de nuevo a un emisor en el invernadero, este al recibirlo modula el ancho de pulso (PWM) entre valores de 1 a 5 volts, este voltaje pasa a través de un convertidor voltaje-corriente que nos proporcionara una entrada para el variador de 4-20 mA, valores con los cuales el variador puede modificar la frecuencia del motor del Airwet.

Tabla VII Valores de % de señal y frecuencia de trabajo

Equivalencia de trabajo en el variador		
Error	% de señal	Frecuencia de trabajo (Hz)
1	36	23.9
3	50	30.7
5	64	38.5
7	79	47.5
8	86	52.8
9	96	58.7
10	100	60

4.8.1 Programación del variador de velocidad

El variador de velocidad AF-60 LP micro drive de la marca general electric, posé una serie de comandos para su programación o un software para realizarlo el software para su programación es el DCT-10, sin embargo, el en nuestro caso el variador fue programado de manera manual. En la figura 4.23 se muestra el variador utilizado

Figura 4.23 Variador AF-60 micro drive General Electric

Para programar el variador de forma manual se debe acceder a el menú de programación desde la pantalla LCD del variador para movernos entre las opciones utilizamos los botones en el panel, en la figura 4.24 se muestra el panel frontal y sus componentes

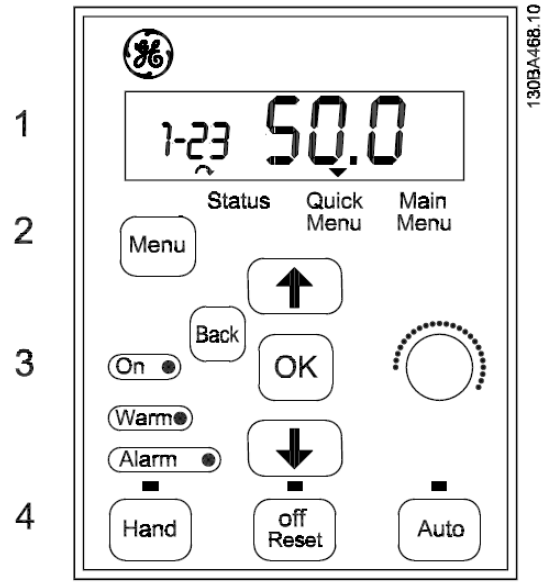


Figura 4.24 Panel frontal (1- Pantalla LCD. 2- Botón Menú, 3- Teclas de navegación. 4-Teclas de funcionamiento y luces indicadors (Led´s))

En el display se puede leer distintos tipos de información como Set up number o número de ajuste , el sentido de giro del motor, los números pequeños muestran el parámetro seleccionado y los grandes el valor del mismo, así como a su derecha se muestra las unidades del parámetro seleccionado, ver figura 4.25.

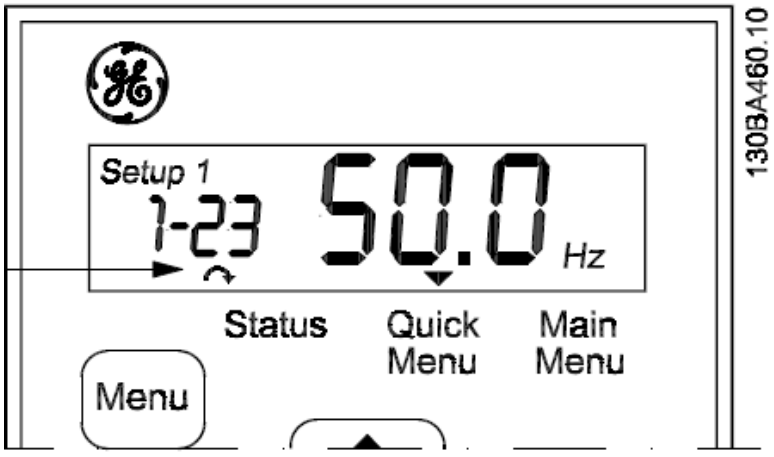


Figura 4.25 Display con opciones desplegadas

La configuración de los parámetros para el control de un motor se pueden llevar a cabo desde el menú rápido o Quick Menu, para ingresar al menu se utiliza la tecla Menu, una vez el indicador este posicionado sobre la opción se usan las teclas de desplazamiento para seleccionar el menú rápido uno o dos y se presiona la tecla OK. Se utilizó el QM1.

Una vez dentro del menú rápido seleccionado se configuran los parámetros de acuerdo a la placa del motor estos parámetros son:

1-20 Potencia del motor [Kw]/[Cv]

1-22 Tensión del motor

1-23 Frecuencia del motor

1-24 Intensidad del motor

1-25 Velocidad del motor

1-29 Auto ajuste

3-02 Referencia mínima

3-03 Referencia máxima

3-41 Intervalo tiempo aceleración

3-42 Intervalo tiempo desaceleración

Con estos datos el variador podrá controlar el motor conectado. Los datos de la placa del motor de la bomba son:

- Voltaje: 220v
- Amperaje: 4.2 A
- Hp: 0.9
- Hz: 60
- RPM: 1700

4.8.2 Convertidor Voltaje-Corriente con salida 4-20 mA

Para controlar el variador de velocidad se requiere una señal de 4-20 mA, esta señal es obtenida gracias al circuito mostrado en la figura 4.26.

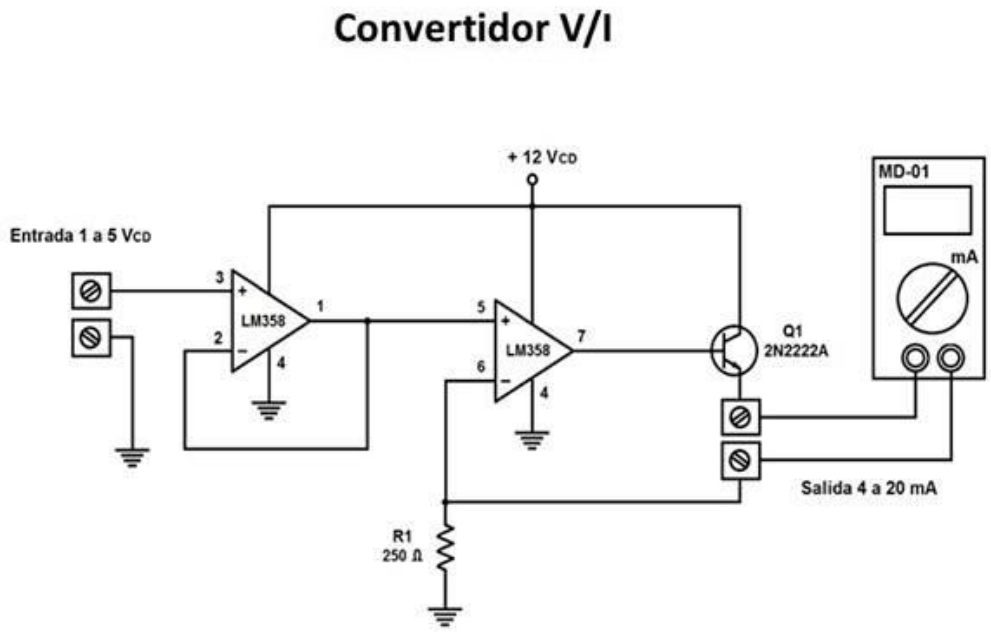


Figura 4.26 Convertidor V/I con salida 4- 20 mA

Este circuito permite la conversión de la señal emitida por el PWM del Arduino que es una señal de 1 a 5 volts a una señal de corriente con valores entre 4 - 20 mA. La señal de salida es conectada a los pines 59 y 60 del variador de velocidad (ver figura 4.27).

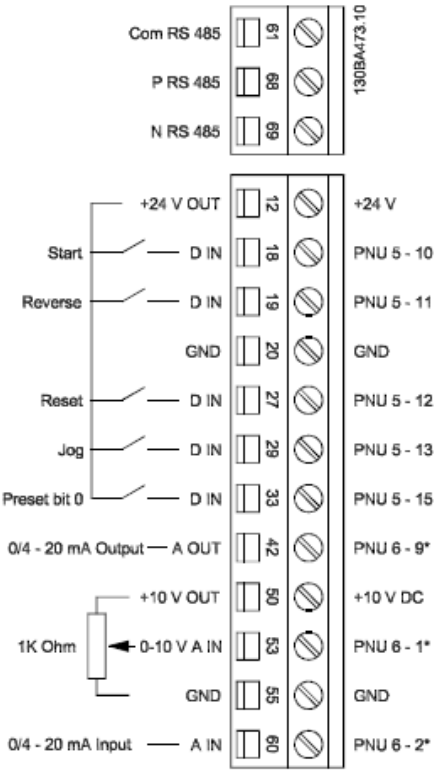


Figura 4.27 Visión general de pines de variador de velocidad AF-60

4.8.3 Configuración de sensores Ds18b20 para la red en invernadero

En la figura 4.21 se muestra la red de 16 sensores Ds18b20 propuesta inicialmente, sin embargo por limitaciones de hardware se tuvo que configurar dicha red para una correcta lectura de los sensores conectados en ella, esta configuración se presenta en la figura 4.28.

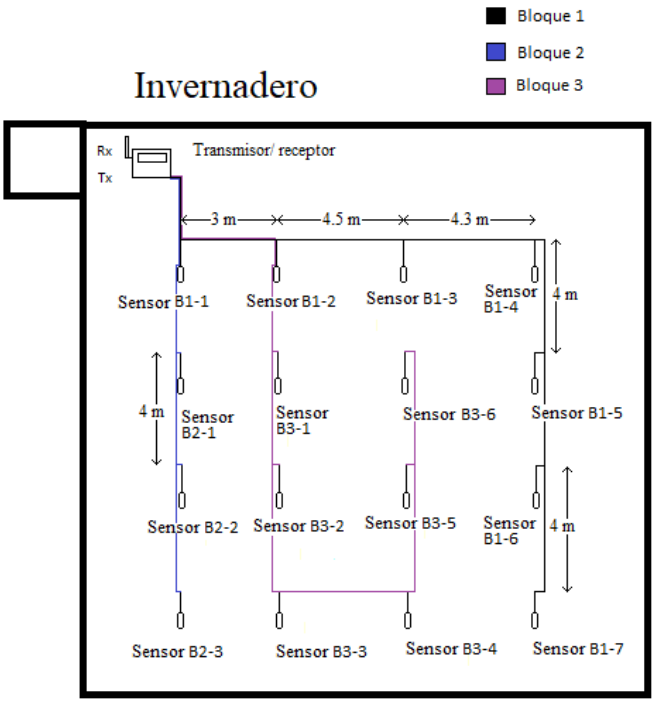


Figura 4.28 Configuración final de los sensores

Se configuraron tres líneas de sensores a las cuales se les llamo bloques, esto permite una correcta adquisición de los datos censados, los cuales posteriormente son enviados al módulo receptor/ emisor para realizar el control. En la figura 4.29 se muestra la conexión del sensor en la red.

Figura 4.29 Sensor montado en la red

4.9 Creación de control PID en RSlogix 500

El programa para el control PID del PLC fue desarrollado en el software RSlogix 500. Con este programa se controla el humificador utilizando un PLC en lugar del control neuronal. En la figura 4.30 se ve parte de su programación. Para realizar este control se realizó un programa de adquisición de datos en Arduino el cual calcula la temperatura promedio y otorgaba una salida de 1-5 v de ancho de pulso (PWM), este voltaje era convertido a corriente con el circuito mostrado en la figura 4.26, dicha entrada analógica es conectada a los pines 59 y 60 del variador.

En la figura 4.30 se muestra el bloque para adquirir y mostrar la temperatura, también es calculado un error en el bloque PID para realizar el control de la variable.

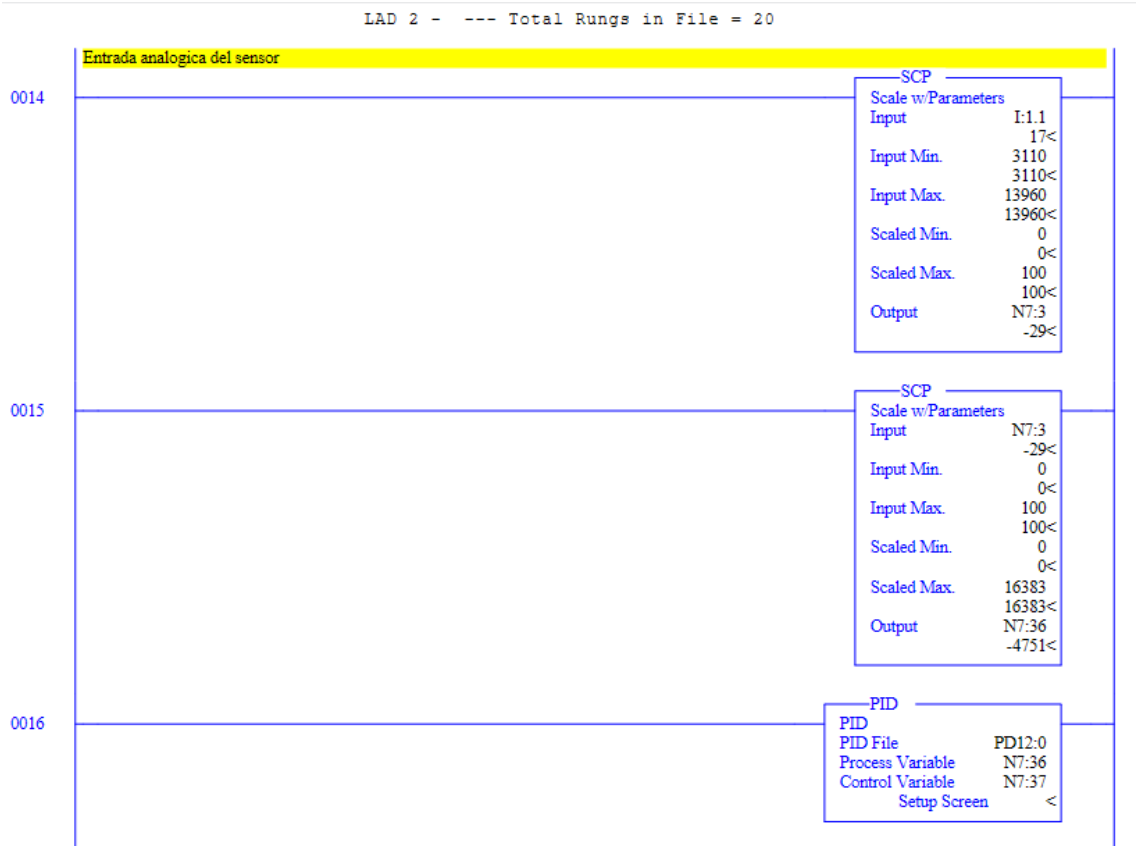


Figura 4.30 Bloques de programación del control en RSlogix 500

Capítulo 5
Experimentos y resultados

5 Experimentos y resultados

5.1 Prueba de velocidad y volumen de aire

La prueba de velocidad se llevo a cabo utilizando un anemómetro digital Benetech Gm8902 (figura 5.1), el resultado de la lectura de velocidad fue de 4.5 mts/s, y su capacidad volumétrica fue de 59.09 mt^3/min, la capacidad, el volumen total del invernadero es de 1000 mt^3, esto quiere decir que se necesitan aproximadamente 16 ventiladores de las mismas características para poder mover el volumen total de aire del invernadero en una hora. Este ventilador se muestra en la figura 5.2, y no es el Airwet.

La prueba se llevó a cabo con el control manual de del variador.

Figura 5.1 Anemómetro digital Benetech Gm8902

Figura 5.2 Ventilador para flujo de aire en el interior del invernadero

5.2 Prueba de red neuronal en el osciloscopio

La red neuronal se probó mediante una salida PWM de un Arduino, para realizar esta prueba de crearon dos instrumentos virtual (Vi) en LabVIEW.

En el primer programa solo se conectó la red neuronal y se simulo la temperatura mediante un control virtual, de esta manera se comprobó la salida PWM a 25, 50, 75 y 100% como se muestra en la tabla VIII.

Tabla VIII Valores de pulso PWM vs voltaje de salida correspondiente

Porcentaje %	Valor pulso (PWM)	Voltaje salida
0	51	1.0000
5	61.2	1.2
10	71.4	1.4
15	81.6	1.6
20	91.8	1.8
25	102	2
30	112.2	2.2
35	122.4	2.4
40	132.6	2.6
45	142.8	2.8
50	153	3
55	163.2	3.2
60	173.4	3.4
65	183.6	3.6
70	193.8	3.8
75	204	4
80	214.2	4.2
85	224.4	4.4
90	234.6	4.6
95	244.8	4.8
100	255	5.0000

El pin utilizado para la salida PWM del Arduino fue el 6, en las figuras 5.3, 5.4, 5.5 y 5.6 se muestran los resultados de la prueba.

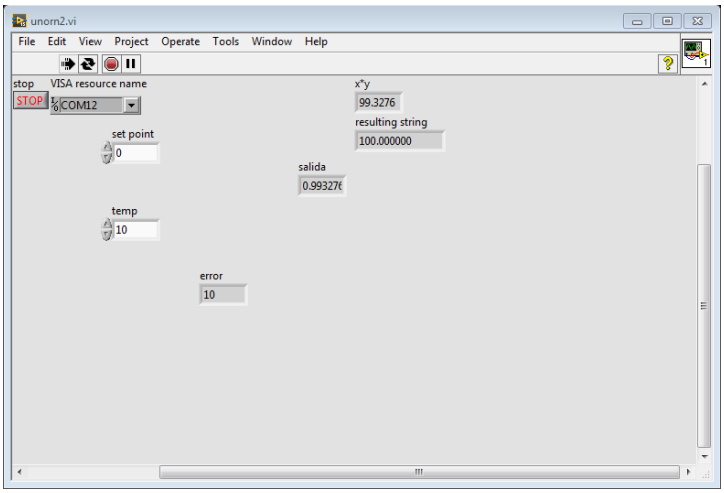


Figura 5.3 Salida a 100%

Figura 5.4 Salida 100% mostrada en el osciloscopio

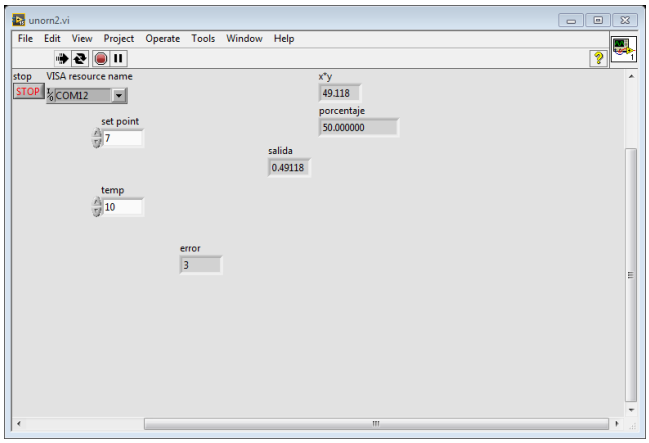


Figura 5.5 Salida a 50% de su capacidad

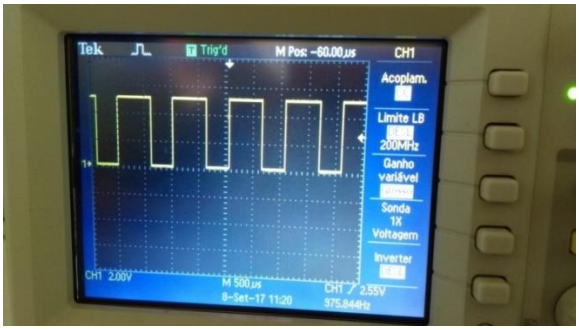


Figura 5.6 Salida a 50% de su capacidad mostrada en osciloscopio

El circuito también se probó con el sensor de temperatura, para verificar que la salida pwm no se viera afectada en condiciones reales, en esta ocasión se utilizaron dos módulos Arduino, uno para realizar la adquisición de datos y el segundo para modular la salida pwm. En la figura 5.7 se muestra la conexión del sensor.

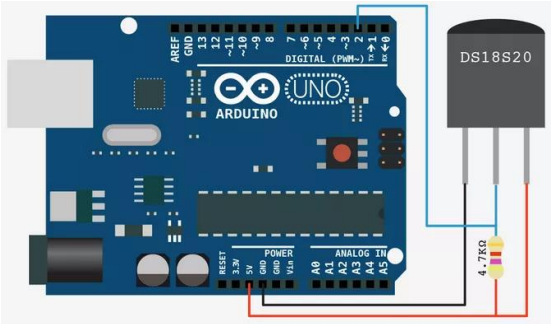


Figura 5.7 Conexión del sensor a la placa Arduino

Los resultados de esta prueba son mostrados en las figura 5.8 y 5.9.

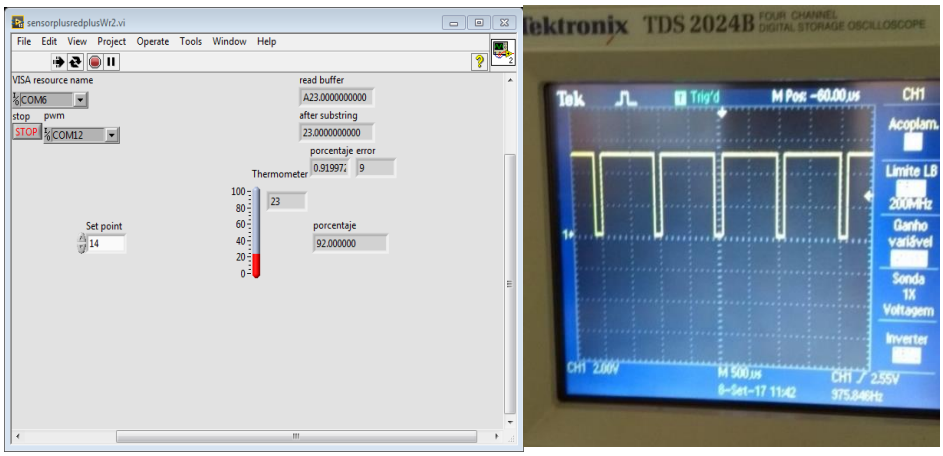


Figura 5.8 Error y porcentaje de salida del PWM a 0%

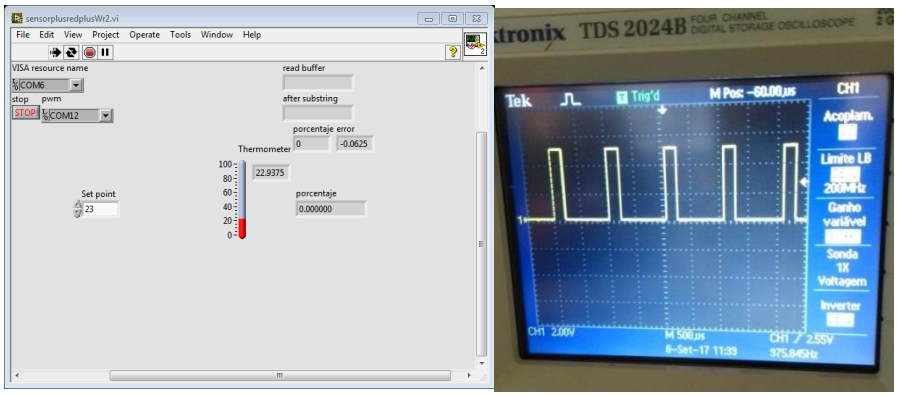


Figura 5.9 Error y porcentaje de salida del pwm 92%= 4.6v

5.3 Prueba del control con red neuronal y ventilador de Airwet

La prueba de control neuronal y el ventilador permitió ver el comportamiento del variador en respuesta a la entrada suministrada por la red neuronal. Esta prueba fue realizada en laboratorio (figura 5.10).

Figura 5.10 Prueba con control neuronal

El experimento consistió en conectar los Arduino y sensores a la computadora y al variador de velocidad que a su vez regularía la frecuencia entrante al ventilador del Ariwet en la figura 5.11 se muestra el lazo de control.

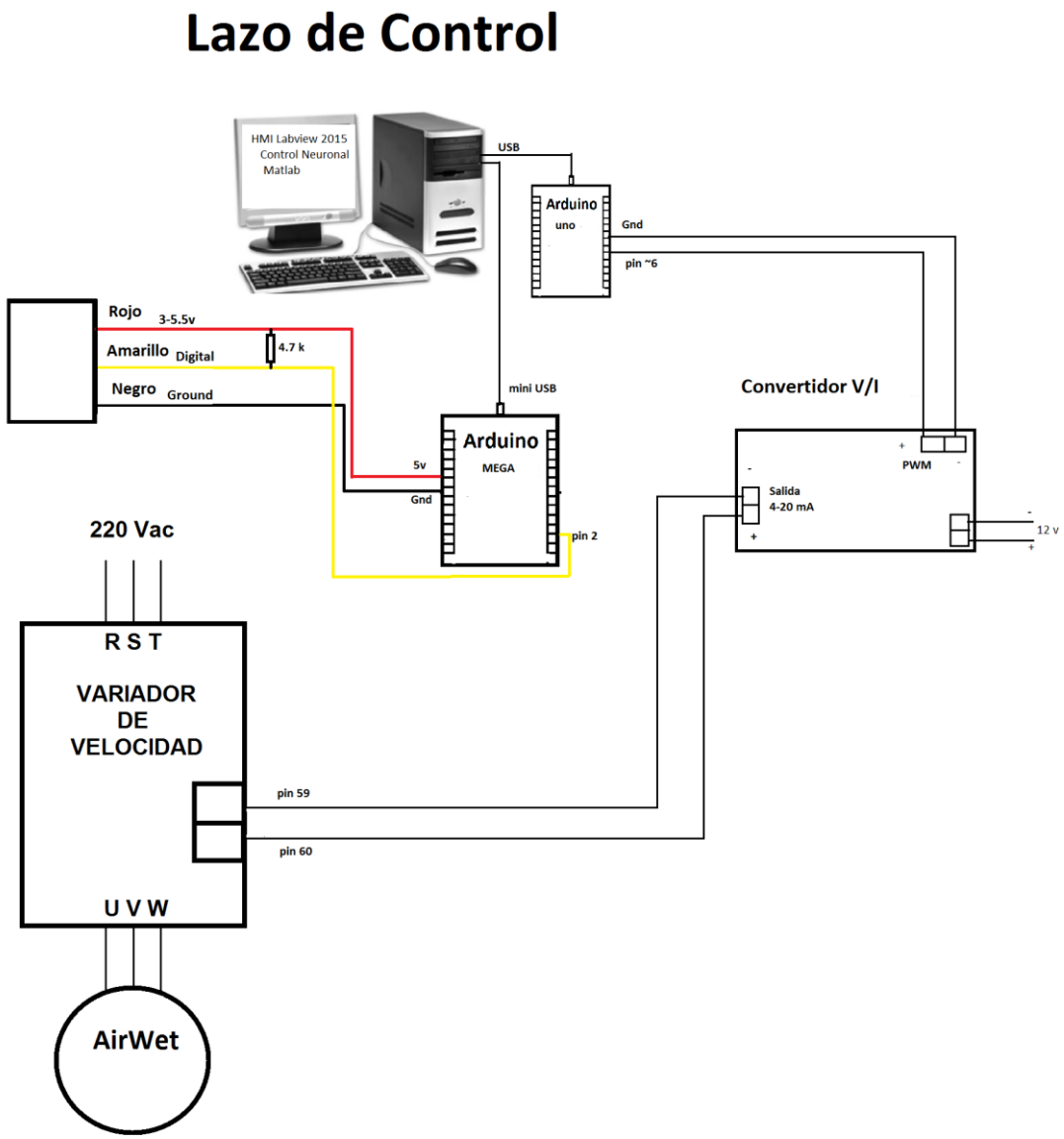


Figura 5.11 Lazo de control del control neuronal

El resultado de este experimento arrojo el siguiente comportamiento en relación al porcentaje 0-100% suministrado por el Arduino a la entrada analógica de del PLC (ver tabla IX)

Tabla IX Relación error-frecuencia entre controlador y variador

Error	Señal (%)	Frecuencia variador (Hz)
1	36	23.9
3	50	30.2
5	64	38.5
7	79	47.5
8	86	52.8
9	96	58.7
10	100	60

5.4 Obtención de Planta para el Control PID del invernadero

Esta prueba se llevó a cabo en el invernadero, en esta ocasión el sensor fue colocado en el centro del invernadero y el Airwet en un extremo como se muestra en la figura 5.12.

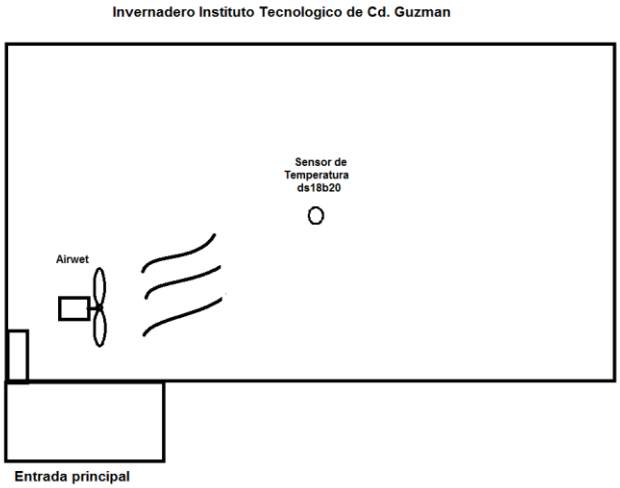


Figura 5.12 Posición del Airwet y sensor de temperatura

Las condiciones en la que se realizó la obtención de la planta fueron las siguientes:

- Cortinas del invernadero abiertas
- Prueba realizada a las 2:00 pm
- Día soleado

La temperatura se monitoreo por tres minutos para comprobar que el sensor estuviese trabajando de manera correcta, posteriormente se encendió el ventilador/humificador del Airwet para observar el comportamiento de la temperatura y cuál sería el rango máximo de que un solo ventilador podría provocar, los resultados se presentan en la gráfica (figura 5.13) que se muestra a continuación, el tiempo de muestrea fue cada 2 min.

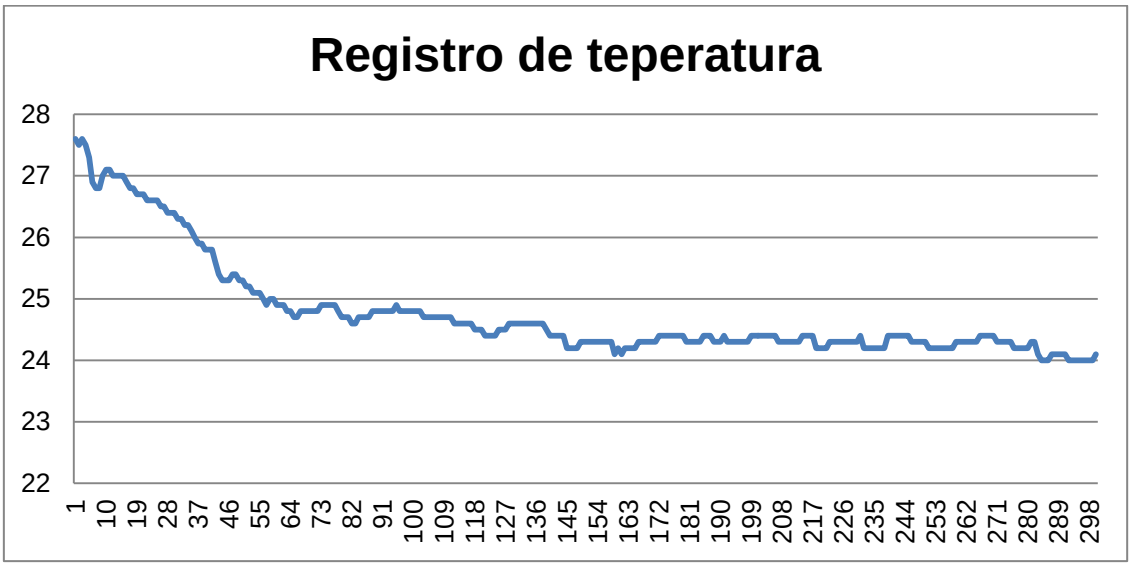


Figura 5.13 Registro de temperatura

Los datos obtenidos con este experimento son utilizados para la obtención de la plata del invernadero, este procedimiento es explicado en el capítulo 3: materiales y métodos.

5.5 Prueba de variación de la temperatura en el invernadero

Esta prueba tuvo la finalidad de verificar la distribución del agua pulverizada con el Airwet y como es el impacto en la temperatura del invernadero. En esta prueba se utilizaron tres sensores distribuidos como se muestra en el diagrama (figura 5.14), teniendo un censado cada cuatro metros.

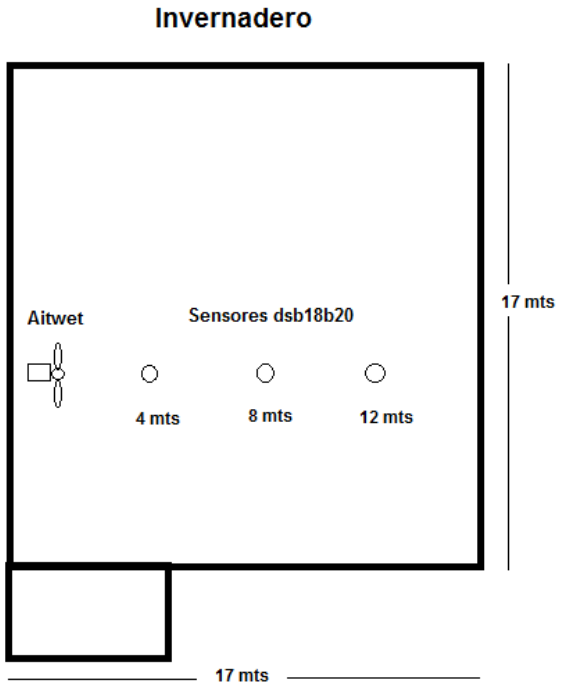


Figura 5.14 Invernadero y distribución de sensores

Los resultados se muestran en la tabla X y tabla XI.

Tabla X Valores iniciales de los sensores

Sensor (4 mts)	Sensor (8 mts)	Sensor (12 mts)
28.4375	28.5	29.1875
28.4375	28.5	29.1875
28.4375	28.5	29.1875
28.4375	28.5	29.1875
28.4375	28.5	29.1875
28.4375	28.5	29.1875
28.4375	28.5	29.1875

Tabla XI Valores finales de los sensores

Sensor (4 mts)	Sensor (8mts)	Sensor (12 mts)
15.6875	16.8125	24.125
15.6875	16.8125	24.125
15.6875	16.8125	24.1875
15.625	16.8125	24.1875
15.625	16.8125	24.1875
15.625	16.8125	24.1875
15.625	16.8125	24.25

Como se observa un Airwet no es suficiente para obtener una repercusión en todo el invernadero, sin embargo dada esta configuración se puede decir que la modulación de la temperatura se apreciara de mayor manera en el sensor que se encuentre a la mitad del flujo de aire del humificador.

5.6 Control neuronal en invernadero

El control neuronal desarrollado en este proyecto fue probado en campo, cabe mencionar que por limitantes de equipo, la prueba fue reducida a un solo segmento del invernadero, este segmento corresponde al bloque 2 de las líneas instaladas para el censado de la temperatura, este bloque se aprecia en la figura 4.28. El elemento que se controló con la modulación de PWM fue el motor de la bomba de agua del Airwet, la placa de motor es mostrada en la figura 5.15.

Figura 5.15 Placa de motor de la bomba de agua del Airwet

Las características indicadas son:

- Voltaje 220
- Amperes 4.2
- Hp 0.9
- Hz 60
- Rpm 1700

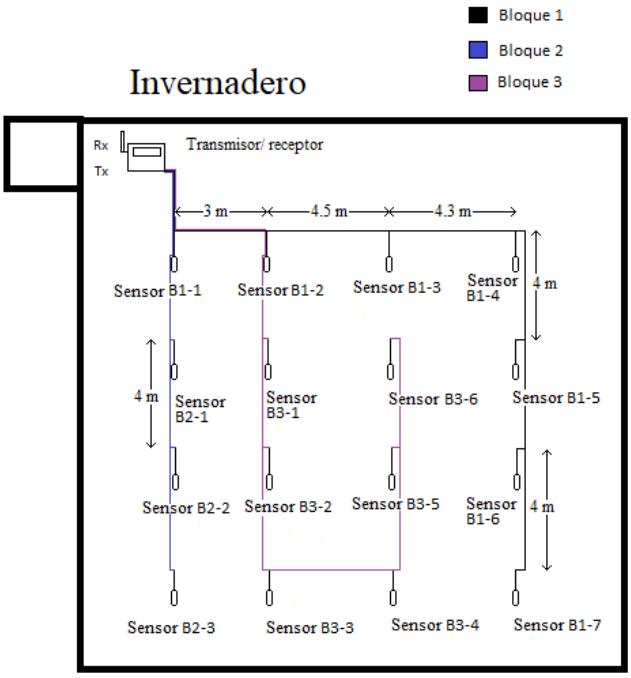


Figura 4.28 Configuración final de los sensores

El control fue realizado con un muestreo de 1000 ms para la temperatura, durante cinco minutos.

La gráfica de control neuronal es mostrada en la figura 5.16, como se puede observar una vez asentado el sistema la temperatura oscila entre uno y dos grados por encima del Set Point establecido.

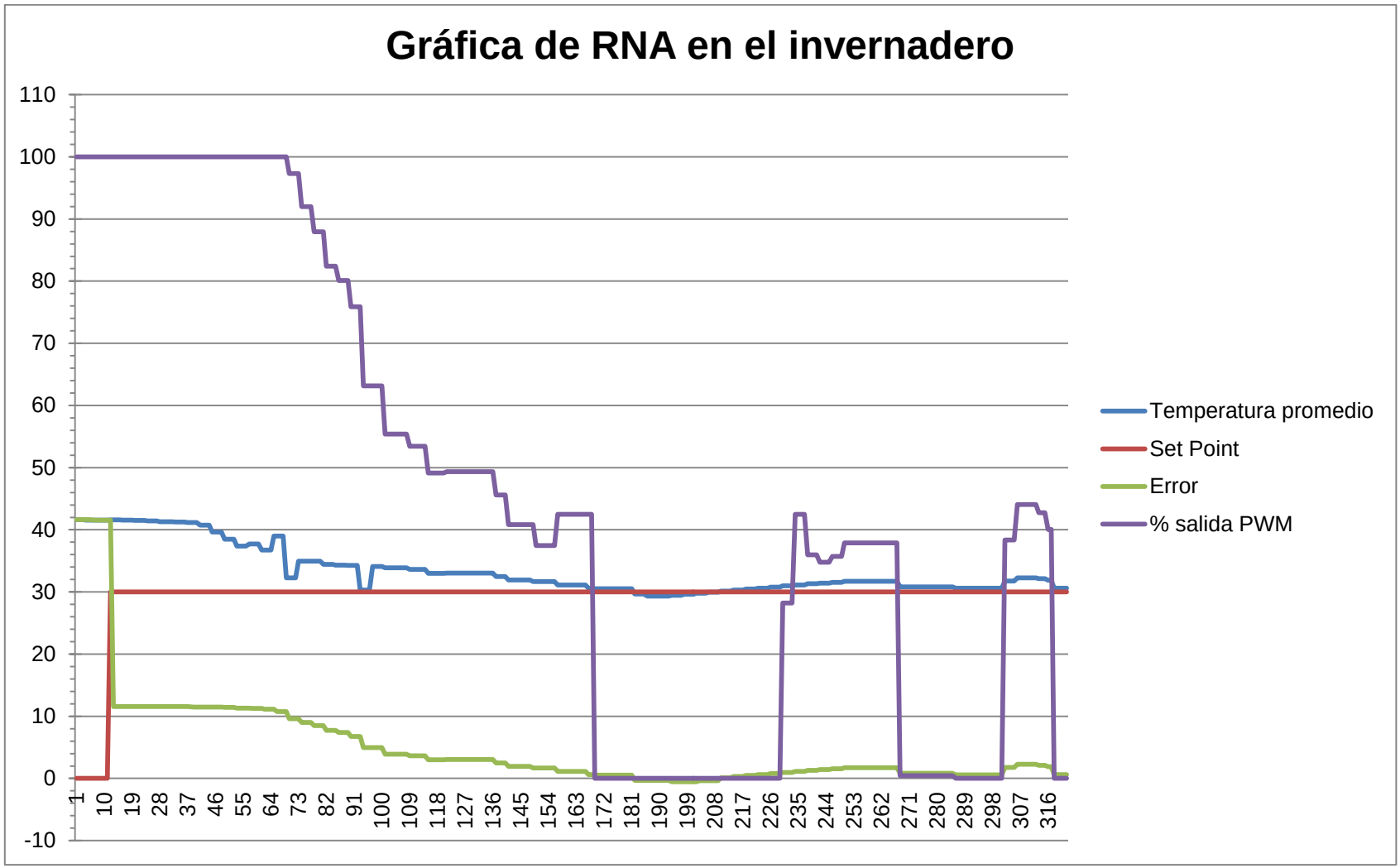


Figura 5.16 Grafica de Control neuronal en el invernadero

El control trabajo de forma óptima y pudo modular la temperatura cerca al Set Point establecido, y como se puede ver en la gráfica la salida mínima al variador de velocidad siempre fue del 35%, lo que nos permitió asegurar que el motor no trabajara de forma errónea o con riesgo de daño.

5.7 Control PID en el invernadero.

El control PID fue probado en campo, para el cálculo inicial de los valores de *Kp*, *Kd*, y *Ki* se utilizó la fórmula de la planta obtenida, esta fórmula fue:

$$G(s) = \frac{1.009}{1 + 45.8s};$$

La cual fue introducida en Matlab y aplicamos la función PID tuner para obtener los valores de los parámetros de control, en la figura 5.17 se muestran los resultados sugeridos

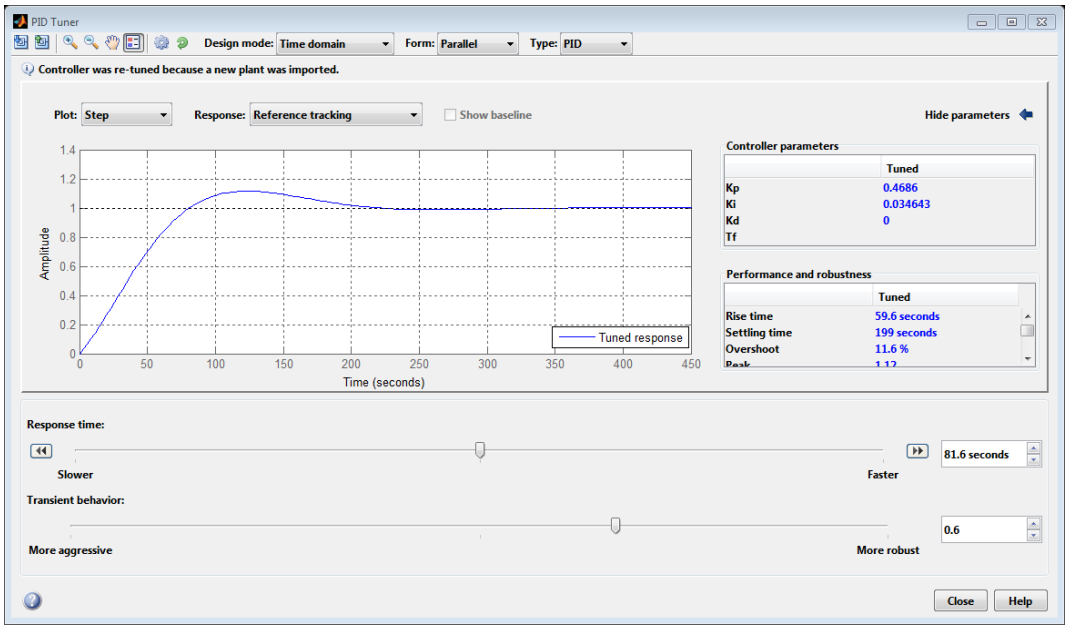


Figura 5.17 Valores sugeridos en para parámetros de control

Sin embargo en la práctica se modificaron dichos resultados para obtener un tiempo de asentamiento más corto, los parámetros obtenidos fueron:

- *Kp*=7
- *Ki*=0.1
- *Kd*=0

El control PID controlo de manera efectiva y logro acercarse al Set Point establecido, esta prueba también se llevó a cabo en el bloque 2 de la red de sensores, debido a que un solo actuador en este caso el humificador Airwet no es suficiente para modular la temperatura del invernadero completo, por tanto la parte afectada no se reflejaría en el resto de red de sensores. En la figura 5.18 se muestra su interface.

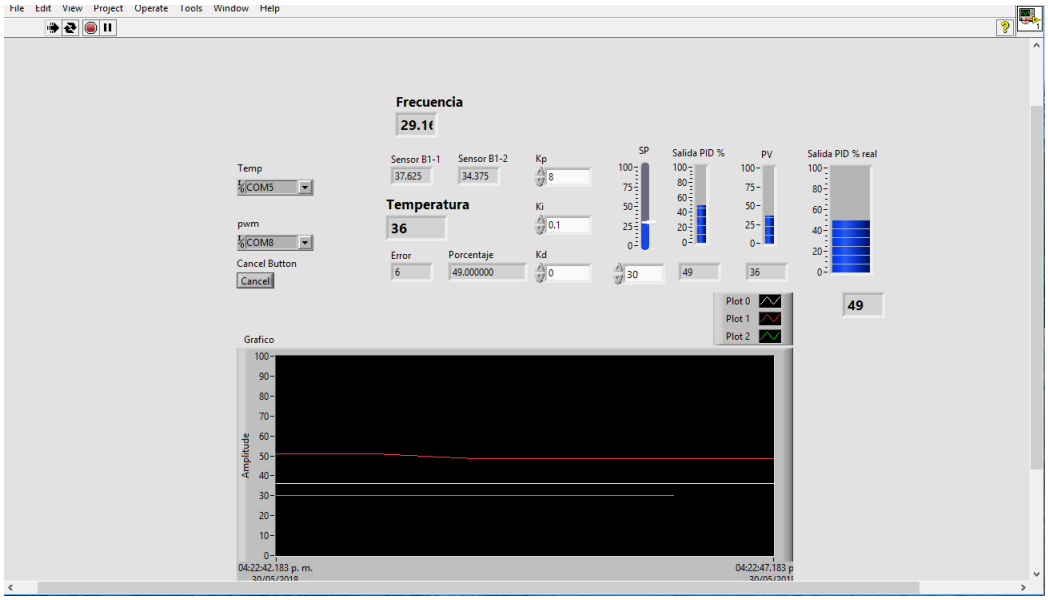


Figura 5.18 Interface control PID

El gráfico de la tabla es presentado en la figura 5.20. Este control presenta una oscilación del Set Point de 1 a 3 grados con respecto a la temperatura deseada. El tiempo de asentamiento fue de 6 minutos.

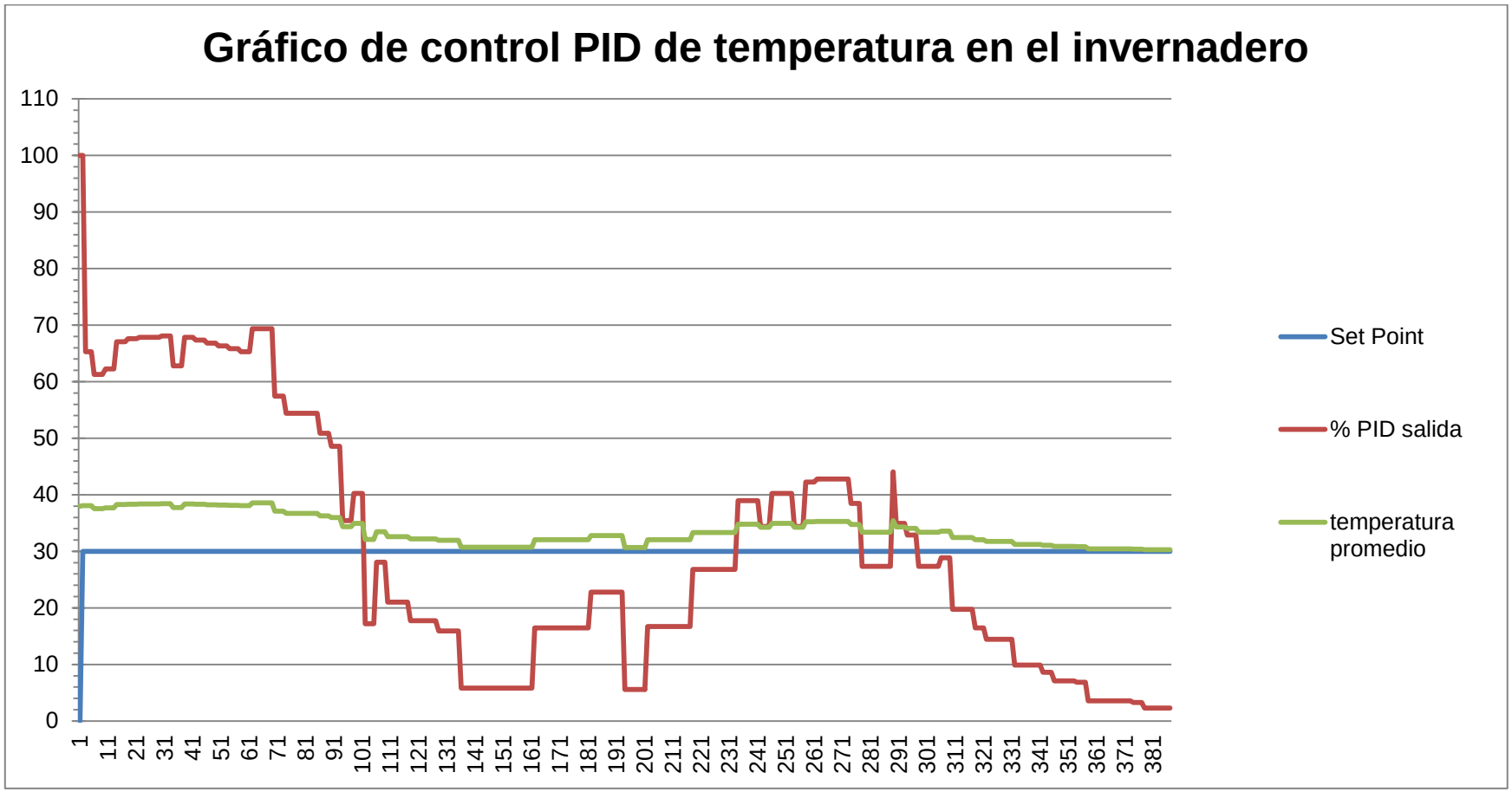


Figura 5.19 Grafica de control PID en el invernadero

Capítulo 6
Conclusiones

6 Conclusiones

6.1 Comparación de controles utilizados en el invernadero

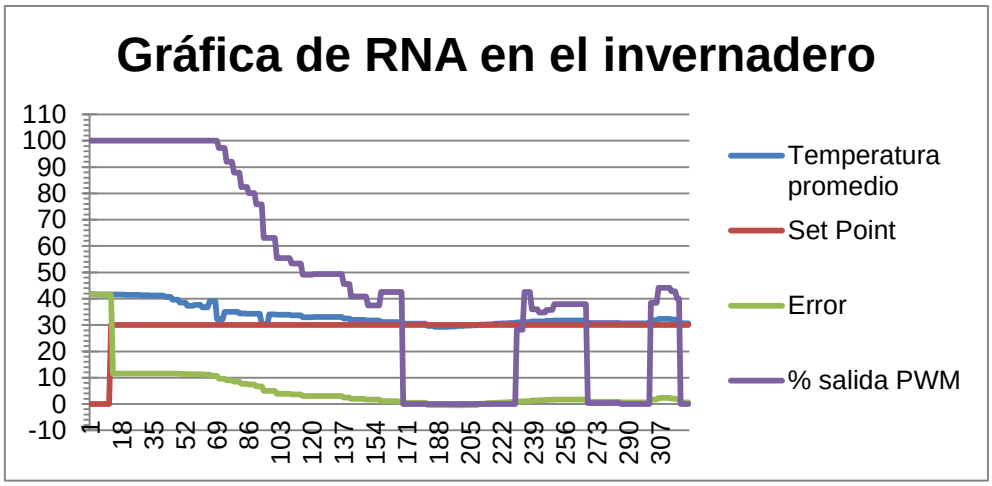


Figura 5.17 Grafica de Control neuronal en el invernadero

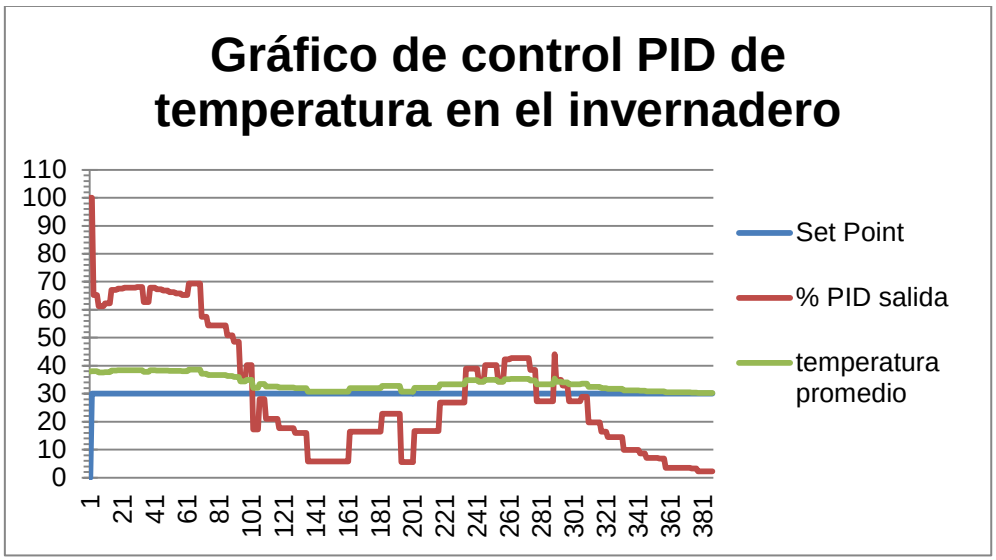


Figura 5.20 Gráfica de control PID en el invernadero

Como se aprecia en las grafica mostradas con anterioridad el control RNA resulto ser más preciso y rápido que el controlador PID, en ambos casos el variador de frecuencia para el motor de la bomba estuvo restringido a 36% para evitar

posibles daños, pero en el control PID la conmutación de la bomba fue más evidente. En el control RNA para mantener la temperatura por un periodo de tiempo solo fue necesario el ventilador, y cuando sobrepasaba el grado de error la red activaba la bomba para volver a estabilizarlo, y como se puede observar en la gráfica del controlador RNA llego a disminuir un grado la temperatura por debajo del set Point.

El controlador de red neuronal tuvo un comportamiento más estable en lo que a señal de salida se refiere, mostrando un incremento y decremento de la señal enviada para modular el PWM de manera más suave.

6.2 Observaciones

Durante el desarrollo de este proyecto se presentaron diferentes situaciones e inconvenientes que vale la pena destacar, estos inconvenientes son listados a continuación.

- Equipo de control insuficiente para mover el volumen de aire total del invernadero.
- Falta de equipo para conmutación de los motores del Airwet
- La comunicación inalámbrica es funcional pero por el tipo de micro controlador empleado puede llegar a presentar un retraso en la señal de control.
- El Arduino uno es un micro controlador útil para diferentes tareas pero en ambientes donde se requiere manejo de alto voltaje puede presentar ruido debido a los componentes acoplados
- El control neuronal es fiable para tareas específicas siempre que se conozcan los valores de entrada y los valores requeridos
- Los controles inteligentes pueden remplazar a los controles clásicos, siempre que las condiciones de uso sean favorables y su implementación no representen un gasto excesivo.
-

6.3 Trabajos a futuro

Los trabajos a futuro empleando un control neuronal son listados a continuación:

- Implementación de sistema de riego automático en base a la humedad del suelo.
- Reconocimiento de enfermedades de los frutos por visión artificial.
- Control de PH en el suelo.

Los trabajos requeridos para el mejoramiento del entorno de trabajo asi como su equipamiento son:

- Mejorar la comunicación inalámbrica empleada para el control de temperatura
- Adquisición de equipo para elementos de control así como instalación eléctrica para su montaje
- Probar el control con diferente micro controlador para señal de control
- Establecer un área segura y asilada para evitar ruido eléctrico en la señal de control.

6.4 Conclusión general

Este proyecto ayudó a demostrar la capacidad que tienen los controles inteligentes para las tareas específicas, demostrado que son viables en estos entornos. Debido a su naturaleza les permite trabajar en condiciones donde no se tiene mucha información o no se puede obtenerla de manera sencilla, como ejemplo se tuvo el controlador PID de este proyecto, que aunque controló de manera similar el método de sintonización y adquisición de la planta fueron un requerimiento extra para su desarrollo, en cambio la creación y desarrollo de la red neuronal se basó en determinar los datos de entrada y su correlación con los datos de salida, y como se comprobó tuvo un rendimiento aceptable al realizar el control requerido. En el anexo 8.7 se muestra un diagrama de conexión sugerido para realizar el control de RNA o PID.

Referencias

REFERENCIAS

[1] Ponce Cruz, P. (2010). Inteligencia artificial con aplicaciones a la ingeniería. 1st ed. CD. de Mexico: Alfaomega Grupo Editor, S.A. de C,V.

[2] Katsuhiko Ogata. (2010). Controladores PID y controladores PID modificados. En Ingeniería de control moderna (567). Madrid: PEARSON EDUCACIÓN

[3] Mandado-Pérez, Enrique; Marcos-Acevedo, Jorge; Fernández-Silva, Celso; Armesto-Quiroga, José-I. Autómatas Programables y sistemas de automatización. México, Marcombo-Alfaomega, 2009, 1120 pag.

[4] Astrom, Karl; Hagglund, Tore. Control PID avanzado. Pearson Prentice Hall. Madrid, España, (2009), 35-312 pag

[5] Creus, Antonio. Instrumentación Industrial. 8A Edición. Marcombo-Alfaomega. México, 2011, 776 pag.

[6] Roca-Cusidó, Alfred. Control de procesos. Barcelona, España. Alfaomega., 1999, 626 pag.

[7] Luyben, W-L. Process Modeling, Simulation, and Control for Chemical Engineers. Mc Graw-Hill, New York, 1973, 558 pag.

[8] Sagarpa.gob.mx. (2018). Superficie Agrícola Protegida. [En línea] Disponible: http://www.sagarpa.gob.mx/quienesomos/datosabiertos/siap/Paginas/superficie_agricola_protegida.aspx [Consultada 4 Ene. 2018].

[9] Digital.ni.com. (2018). ¿Que es una Señal Modulada por Ancho de Pulso (PWM) y Para Qué es Utilizada? - National Instruments. [En Línea] Disponible: http://digital.ni.com/public.nsf/allkb/AA1BDEA4AA224E3E86257CE400707527 [Consultada 11 May 2018].

[10] Arduino UTFSM. (2018). Modulación por ancho de pulso (PWM). [En Línea] Disponible: http://www.arduino.utfsm.cl/modulacion-por-ancho-de-pulso-pwm/ [Consultada 11 May 2018].

[11] Posada Contreras, Johnny. (25, julio-diciembre, 2005). Modulación por ancho de pulso (PWM) y modulación vectorial (SVM). El Hombre y la Máquina, 25, 70-83. 11 mayo 2018, De http://www.redalyc.org/articulo.oa?id=47802507 Base de datos. [Consultada 12 May 2018].

[12] Electrónica Unicrom. (2018). Convertidor Analógico Digital. CAD - ADC - Electrónica Unicrom. [En línea] Disponible: https://unicrom.com/convertidor-analogico-digital-cad-adc/ [Consultada 4 May 2018].

[13] Agrobit.com. (2018). Tipos de invernaderos. [En Línea] Disponible: http://www.agrobit.com/Info_tecnica/alternativos/horticultura/AL_000010ho.htm [Consultada 4 mayo 2018].

[14] Logicelectronic.com. (2018). Diferencias entre PAC y PLC. [En línea] Disponible en: http://www.logicelectronic.com/BECKHOFF/Que%20es%20un%20PAC.html [Consultada 4 Abr. 2018].

Anexos

ANEXOS

Anexo 1: Programa Arduino Emisor A (invernadero)

Programa encargado de tomar lecturas de los sensores y enviarlas al cuarto de control.

```
 // Incluimos las librerías necesarias
  //*****************************************radio************
  #include <SPI.h> // Librería para la comunicación SPI
  // Librerías para el funcionamiento del módulo NRF24L01
  #include <nRF24L01.h>
  #include <RF24.h>

  // Declaramos los pines de control del módulo NRF24L01
  #define CE 9
  #define CSN 10


  // libreiras sensor Dsb20
#include <OneWire.h>
#include <DallasTemperature.h>
//*****librerias i2c*****
#include <LiquidCrystal_I2C.h> // Debe descargar la Libreria que controla el I2C
#include<Wire.h>
#include <LCD.h>




  // Se crea el objeto tipo RF24
  RF24 radio(CE, CSN);

  // Se declaran los canales (64 bits en hexadecimal) para transmisión RF
  //const uint64_t canal[2] = {0xF0F0F0F0E1LL,0xF0F0F0F0D2LL};
  const byte direccion0[6]= {'l','i','n','e','a','1'};
 // const byte direccion1[6]= {'l','i','n','e','a','2'};

  // Variable que enviamos mediante RF (de tipo string siempre)
int vector[17];

  // Variable que recibimos mediante RF (de tipo string siempre)
  //int PwmSig[1];

 int uno;
 int dos;

// ********* configuracion del sentosr de temperatura**********

OneWire ourWire(2);                //Se establece el pin 2  como bus OneWire
```

```
OneWire ourWire2(3);
OneWire ourWire3(4);

DallasTemperature sensors(&ourWire); //Se declara una variable u objeto para
nuestro sensor
DeviceAddress address1  = {0x28, 0xFF, 0x48, 0xE7, 0x62, 0x16, 0x4,
0x9D};//dirección del sensor 1
DeviceAddress address2  = {0x28, 0xFF, 0x24, 0x2F, 0x21, 0x16, 0x5,
0x3};//dirección del sensor 2
DeviceAddress address3  = {0x28, 0xFF, 0x5E, 0xDC, 0x62, 0x16, 0x4,
0xA1};//dirección del sensor 3
DeviceAddress address4  = {0x28, 0xFF, 0x65, 0xE7, 0x62, 0x16, 0x4,
0x89};//dirección del sensor 4
DeviceAddress address5  = {0x28, 0xFF, 0x6D, 0xB2, 0x62, 0x16, 0x4,
0xEE};//dirección del sensor 5
DeviceAddress address6  = {0x28, 0xFF, 0x47, 0xE7, 0x62, 0x16, 0x4,
0xF2};//dirección del sensor 6
DeviceAddress address7  = {0x28, 0xFF, 0x77, 0xC3, 0x62, 0x16, 0x4,
0x8};//dirección del sensor 7


DallasTemperature sensors2(&ourWire2);
DeviceAddress address8  = {0x28, 0xFF, 0xA4, 0xE7, 0x62, 0x16, 0x4,
0x66};//dirección del sensor 8
DeviceAddress address9  = {0x28, 0xFF, 0x5F, 0xD4, 0x62, 0x16, 0x4,
0x70};//dirección del sensor 9

DallasTemperature sensors3(&ourWire3);
DeviceAddress address10 = {0x28, 0xFF, 0x50, 0xC4, 0x62, 0x16, 0x4,
0x27};//dirección del sensor 10
DeviceAddress address11 = {0x28, 0xFF, 0x4A, 0xB2, 0x62, 0x16, 0x4,
0x47};//dirección del sensor 11
DeviceAddress address12 = {0x28, 0xFF, 0x5A, 0xE6, 0x62, 0x16, 0x4,
0xED};//dirección del sensor 12
DeviceAddress address13 = {0x28, 0xFF, 0x3A, 0xD9, 0x62, 0x16, 0x4,
0x2E};//dirección del sensor 13
DeviceAddress address14 = {0x28, 0xFF, 0x8D, 0xC8, 0x62, 0x16, 0x4,
0x87};//dirección del sensor 14
DeviceAddress address15 = {0x28, 0xFF, 0x1B, 0xD2, 0x62, 0x16, 0x4,
0x8F};//dirección del sensor 15
//DeviceAddress address16 = {0x28, 0xFF, 0x4B, 0x98, 0x73, 0x15, 0x2,
0x88};//dirección del sensor 16


//********* pantalla LCD con i2c

#define I2C_ADDR    0x27

#define  LED_OFF  0
#define  LED_ON  1
//mjkdz i2c LCD board
//                   addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C             lcd(I2C_ADDR,2, 1, 0, 4, 5, 6, 7);
```

```
//LiquidCrystal_I2C            lcd(I2C_ADDR,4, 5, 6, 0, 1, 2, 3, 7, NEGATIVE);


//******contador y promedio
int conta;   // contador de sensores activos
float tprom; // temperatura promedio calculo
//*************


  void setup()
  {
     Serial.begin(9600);
      radio.begin();  // Inicialización de la comunicación RF
      sensors.begin();   //Se inicia el sensor de temperatura
      sensors2.begin();
      sensors3.begin();

      // Establece el retardo y el número de reintentos tras fallo en la
comunicación RF
      radio.setRetries(15,15);
      radio.stopListening(); // Paro de escuchar por el canal "1"
      radio.openWritingPipe(direccion0); // Abro el canal "0" para escribir
     // radio.openReadingPipe(1,direccion1); // Abro el canal "1" para leer
      //lcd
  //*** pantalla lcd
 lcd.begin (16,2);    // Inicializar el display con 16 caraceres 2 lineas
 lcd.setBacklightPin(3,POSITIVE);
 lcd.setBacklight(LED_ON);
 // lcd.init();
 // lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temperatura"); // Mensaje a despegar
  delay(2500);
   lcd.clear();


  }

  void loop()
  {

/// adquisision de datos con sensores
conta=0;

   sensors.requestTemperatures();   //envía el comando para obtener las
temperaturas
   sensors2.requestTemperatures();
   sensors3.requestTemperatures();
  //******************
float temp1= sensors.getTempC(address1);//Se obtiene la temperatura en °C del
sensor 1
if(temp1>=0){
```

```
  conta=conta+1;
  }
  else{temp1=0;}
float temp2= sensors.getTempC(address2);//Se obtiene la temperatura en °C del
sensor 2
if(temp2>=0){
  conta=conta+1;
  }
  else{temp2=0;}
float temp3= sensors.getTempC(address3);//Se obtiene la temperatura en °C del
sensor 3
if(temp3>=0){
  conta=conta+1;
  }
 else{temp3=0;}
//**************************************

float temp4= sensors.getTempC(address4);//Se obtiene la temperatura en °C del
sensor 3
if(temp4>=0){
  conta=conta+1;
  }
  else{temp4=0;}


float temp5= sensors.getTempC(address5);//Se obtiene la temperatura en °C del
sensor 3
if(temp5>=0){
  conta=conta+1;
  }
  else{temp5=0;}

  float temp6= sensors.getTempC(address6);//Se obtiene la temperatura en °C del
sensor 3
if(temp6>=0){
  conta=conta+1;
  }
  else{temp6=0;}

  float temp7= sensors.getTempC(address7);//Se obtiene la temperatura en °C del
sensor 3
if(temp7>=0){
  conta=conta+1;
  }
  else{temp7=0;}

  float temp8= sensors2.getTempC(address8);//Se obtiene la temperatura en °C del
sensor 3
if(temp8>=0){
  conta=conta+1;
  }
  else{temp8=0;}
```

```
  float temp9= sensors2.getTempC(address9);//Se obtiene la temperatura en °C del
sensor 3
if(temp9>=0){
  conta=conta+1;
  }
  else{temp9=0;}

  float temp10= sensors3.getTempC(address10);//Se obtiene la temperatura en °C del
sensor 3
if(temp10>=0){
  conta=conta+1;
  }
  else{temp10=0;}

  float temp11= sensors3.getTempC(address11);//Se obtiene la temperatura en °C del
sensor 3
if(temp11>=0){
  conta=conta+1;
  }
  else{temp11=0;}

  float temp12= sensors3.getTempC(address12);//Se obtiene la temperatura en °C del
sensor 3
if(temp12>=0){
  conta=conta+1;
  }
  else{temp12=0;}

  float temp13= sensors3.getTempC(address13);//Se obtiene la temperatura en °C del
sensor 3
if(temp13>=0){
  conta=conta+1;
  }
  else{temp13=0;}

  float temp14= sensors3.getTempC(address14);//Se obtiene la temperatura en °C del
sensor 3
if(temp14>=0){
  conta=conta+1;
  }
  else{temp14=0;}

  float temp15= sensors3.getTempC(address15);//Se obtiene la temperatura en °C del
sensor 3
if(temp15>=0){
  conta=conta+1;
  }
  else{temp15=0;}

  if(conta>0){

tprom=(temp1+temp2+temp3+temp4+temp5+temp6+temp7+temp8+temp9+temp10+temp11+ temp12
+ temp13+temp14+temp15)/conta;
```

```
  }
*******************************************

delay(10);

 vector[0]=(int)tprom; //
     vector[1]=(int)temp1;
     vector[2]=(int)temp2;
     vector[3]=(int)temp3;
     vector[4]=(int)temp4;
     vector[5]=(int)temp5;
     vector[6]=(int)temp6;
     vector[7]=(int)temp7;
     vector[8]=(int)temp8;
     vector[9]=(int)temp9;
     vector[10]=(int)temp10;
     vector[11]=(int)temp11;
     vector[12]=(int)temp12;
     vector[13]=(int)temp13;
     vector[14]=(int)temp14;
     vector[15]=(int)temp15;

// Se envía el valor de la variable estado_pulsador_A[0] a través de RF

bool ok=radio.write(vector, sizeof(vector));
      delay(100); // Doy tiempo de escritura al emisor A
      if(ok==1){Serial.println("enviado");}

//LCD
  lcd.setCursor(0, 0);
  lcd.print("Temperatura ");
  lcd.setCursor(0, 1);
   lcd.print("               ");
  lcd.setCursor(0, 1);
  lcd.print(tprom);
  lcd.print(" cen");


delay(1500);
  }
```

Anexo 2: Programa Arduino Receptor A (cuarto de control)

Programa encargado de recibir las lecturas de la temperatura y enviarlas a la interface de LabVIEW para aplicar control neuronal.

```
// Incluimos las librerías necesarias
  #include <SPI.h> // Librería para la comunicación SPI
  // Librerías para el funcionamiento del módulo NRF24L01
  #include <nRF24L01.h>
  #include <RF24.h>

  // Declaramos los pines de control del módulo NRF24L01
  #define CE 9
  #define CSN 10
 //*****librerias i2c*****
  #include <LiquidCrystal_I2C.h> // Debe descargar la Libreria que controla el I2C
  #include<Wire.h>
  #include <LCD.h>

//********* pantalla LCD con i2c

#define I2C_ADDR    0x27

#define  LED_OFF  0
#define  LED_ON  1
//mjkdz i2c LCD board
//                    addr, en,rw,rs,d4,d5,d6,d7,bl,blpol
LiquidCrystal_I2C             lcd(I2C_ADDR,2, 1, 0, 4, 5, 6, 7);
//LiquidCrystal_I2C             lcd(I2C_ADDR,4, 5, 6, 0, 1, 2, 3, 7, NEGATIVE);


  // ************  Se crea el objeto tipo RF24
  RF24 radio(CE, CSN);

  // Se declaran los canales (64 bits en hexadecimal) para transmisión RF
  const byte direccion0[6]= {'1','i','n','e','a','1'};
   const byte direccion1[7]= {'1','2','3','4','5','6','7'};

  // Variable que enviamos mediante RF (de tipo string siempre)
 int PwmSig[1];

  // Variable que recibimos mediante RF (de tipo string siempre)
 int vector[17];
 int temp[17];
 int dos;

 //***** Pwm para plc****

char cadena[30]; //Creamos un array que almacenará los caracteres que escribiremos
en la consola del PC. Le asignamos  un tope de caracteres, en este caso 30
byte posicion=0;  //Variable para cambiar la posición de los caracteres del array
int valor;  //Variable del valor entero
```

```
int pwm = 6; // pin para pwm
float Spwm;
float datas=0;
//********************

// variables de temperatura ***

int T0=0;
int T1=0;
int T2=0;
int T3=0;
int T4=0;
int T5=0;
int T6=0;
int T7=0;
int T8=0;
int T9=0;
int T10=0;
int T11=0;
int T12=0;
int T13=0;
int T14=0;
int T15=0;
int T16=0;

//*****

  void setup()
  {



      Serial.begin(9600);
      radio.begin();  // Inicialización de la comunicación RF

      // Establece el retardo y el número de reintentos tras fallo en la
comunicación RF
      radio.setRetries(15,15);

     // radio.openWritingPipe(direccion1); // Abro el canal "1" para escribir
      radio.openReadingPipe(1,direccion0); // Abro el canal "0" para leer
       radio.startListening();
 //*** pantalla lcd
 lcd.begin (16,2);    // Inicializar el display con 16 caraceres 2 lineas
 lcd.setBacklightPin(3,POSITIVE);
 lcd.setBacklight(LED_ON);
 // lcd.init();
 // lcd.backlight();
  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("Temperatura"); // Mensaje a despegar
```

```
  delay(2500);
   lcd.clear();

  }

  void loop()
  {

      // ********************** PARTE CORRESPONDIENTE A LA RECEPCIÓN B
**************************

    //  radio.startListening(); // Comienzo a escuchar por el canal "0"

      // Siempre que haya información disponible vía RF...
      while(radio.available())
      {
        Serial.println("escucho");
        // Se recibe el valor de la variable estado_pulsador_A[0] a través de RF
        radio.read(vector, sizeof(vector));
       // dos=vector[0];
        //Serial.println(vector[0]);
        ////Serial.println(vector[14]);
         //Serial.println(vector[15]);
         // Serial.println(vector[16]);
      }
      delay(50); // Doy tiempo de lectura al receptor B

       delay(50);
     T0=vector[0];
     T1=vector[1];
     T2=vector[2];
     T3=vector[3];
     T4=vector[4];
     T5=vector[5];
     T6=vector[6];
     T7=vector[7];
     T8=vector[8];
     T9=vector[9];
     T10=vector[10];
     T11=vector[11];
     T12=vector[12];
     T13=vector[13];
     T14=vector[14];
     T15=vector[15];
     T16=vector[16];

//LCD
  lcd.setCursor(0, 0);
  lcd.print("Temperatura ");
  lcd.setCursor(0, 1);
   lcd.print("               ");
  lcd.setCursor(0, 1);
  lcd.print(T0);
```

```
lcd.print(" cen");


  String t0= String(T0,DEC);
  String st0= String("A"+t0);
  Serial.println(st0);

  String t1= String(T1,DEC);
  String st1= String("b"+t1);
  Serial.println(st1);

  String t2= String(T2,DEC);
  String st2= String("c"+t2);
  Serial.println(st2);

  String t3= String(T3,DEC);
  String st3= String("d"+t3);
  Serial.println(st3);

  String t4= String(T4,DEC);
  String st4= String("e"+t4);
  Serial.println(st4);

  String t5= String(T5,DEC);
  String st5= String("f"+t5);
  Serial.println(st5);

  String t6= String(T6,DEC);
  String st6= String("g"+t6);
  Serial.println(st6);

  String t7= String(T7,DEC);
  String st7= String("h"+t7);
  Serial.println(st7);

  String t8= String(T8,DEC);
  String st8= String("i"+t8);
  Serial.println(st8);

  String t9= String(T9,DEC);
  String st9= String("j"+t9);
  Serial.println(st9);

  String t10= String(T10,DEC);
  String st10= String("k"+t10);
  Serial.println(st10);

  String t11= String(T11,DEC);
  String st11= String("l"+t11);
  Serial.println(st11);

  String t12= String(T12,DEC);
  String st12= String("m"+t12);
```

```
    Serial.println(st12);

    String t13= String(T13,DEC);
    String st13= String("n"+t13);
    Serial.println(st13);

    String t14= String(T14,DEC);
    String st14= String("o"+t14);
    Serial.println(st14);

    String t15= String(T15,DEC);
    String st15= String("p"+t15);
    Serial.println(st15);

    String t16= String(T16,DEC);
    String st16= String("q"+t16);
    Serial.println(st16);

delay(1500);

  }
```

Anexo 3: Programa Arduino Emisor B (cuarto de control)

Programa encargado del envió del valor PWM para el control del variador

```
// Incluimos las librerías necesarias
  #include <SPI.h> // Librería para la comunicación SPI
  // Librerías para el funcionamiento del módulo NRF24L01
  #include <nRF24L01.h>
  #include <RF24.h>

  // Declaramos los pines de control del módulo NRF24L01
  #define CE 9
  #define CSN 10

  // Se crea el objeto tipo RF24
  RF24 radio(CE, CSN);

  // Se declara los canal (64 bits en hexadecimal) para transmisión RF

  const byte direccion1[7]= {'1','2','3','4','5','6','7'};

  //***** Pwm para plc****

char cadena[30]; //Creamos un array que almacenará los caracteres que escribiremos
en la consola del PC. Le asignamos  un tope de caracteres, en este caso 30
byte posicion=0;  //Variable para cambiar la posición de los caracteres del array
int valor;  //Variable del valor entero

int pwm = 6; // pin para pwm
float Spwm;
float datas=0;
//********************

int PwmSig[1];

void setup() {
  Serial.begin(9600);
      radio.begin();  // Inicialización de la comunicación RF

      // Establece el retardo y el número de reintentos tras fallo en la
comunicación RF
      radio.setRetries(15,15);
      radio.stopListening(); // Paro de escuchar por el canal "1"
      radio.openWritingPipe(direccion1); // Abro el canal "0" para escribir

}

void loop() {

//***** PWM ******
 if(Serial.available()) //Nos dice si hay datos dentro del buffer
  {
```

```
    memset(cadena, 0,sizeof(cadena));//memset borra el contenido del array
"cadena" desde la posición 0 hasta el final sizeof

    while(Serial.available()>0) //Mientras haya datos en el buffer ejecuta la
función
    {
      delay(5); //Poner un pequeño delay para mejorar la recepción de datos
      cadena[posicion]=Serial.read();//Lee un carácter del string "cadena" de la
"posicion", luego lee el siguiente carácter con "posicion++"
      posicion++;
    }

    valor=atoi(cadena);//Convertimos la cadena de caracteres en enteros
    //Serial.println(valor);//Imprimimos el valor sumandole un valor +2
    posicion=0;//Ponemos la posicion a 0
  }
 else{
     // valor=70;
     }
  delay (50);

 // ******modulacion del pwm para señal de control

      if(valor > 99)
    {
      Spwm=255;
      //Serial.println(Spwm);
      delay (10);
      }
    if (valor>=30 && valor < 100)
    {
     datas=valor*.01;
     Spwm=204*datas+51;
     //Spwm=255;
    // Serial.println(Spwm);
     //Serial.println(datas);
     delay (10);
      }
     if(valor < 30){Spwm=45;
            //Serial.println(Spwm);
            }

           // Serial.println(Spwm);
   //analogWrite(pwm, Spwm);
PwmSig[0]=Spwm;
     Serial.println(Spwm);

     // ******* PARTE CORRESPONDIENTE A LA EMISIÓN B **************************

      // Se envía el valor de la variable estado_pulsador_B[0] a través de RF
      radio.write(PwmSig, sizeof(PwmSig));
      delay(20); // Doy tiempo de escritura al emisor B
}
```

Anexo 4: Programa Arduino Receptor B

Programa encargado de recibir la señal PWM para regular la salida de voltaje y convertirla a corriente, y de esta manera controlar la señal de entrada al variador.

```
// Incluimos las librerías necesarias
  #include <SPI.h> // Librería para la comunicación SPI

  // Librerías para el funcionamiento del módulo NRF24L01
  #include <nRF24L01.h>
  #include <RF24.h>

  // Declaramos los pines de control del módulo NRF24L01
  #define CE 9
  #define CSN 10

  // Se crea el objeto tipo RF24
  RF24 radio(CE, CSN);

  // Se declaran los canales (64 bits en hexadecimal) para transmisión RF

  const byte direccion1[7]= {'1','2','3','4','5','6','7'};

  // Variable que enviamos mediante RF (de tipo string siempre)
 int PwmSig[1];

  // Variable que recibimos mediante RF (de tipo string siempre)
 //int vector[17];


 int uno;

//***** Pwm para plc****

//char cadena[30]; //Creamos un array que almacenará los caracteres que
escribiremos en la consola del PC. Le asignamos  un tope de caracteres, en este
caso 30
//byte posicion=0;  //Variable para cambiar la posición de los caracteres del
array
int valor;  //Variable del valor entero

int pwm = 6; // pin para pwm
float Spwm;
float datas=0;
//********************


  void setup()
  {
```

```
      Serial.begin(9600);
      radio.begin();  // Inicialización de la comunicación RF

// Establece el retardo y el número de reintentos tras fallo en la comunicación RF
      radio.setRetries(15,15);
      radio.openReadingPipe(1,direccion1); // Abro el canal "0" para leer
       radio.startListening(); // Comienzo a escuchar por el canal "1"
  }

  void loop()
  {

   // delay(100);

  // ******* PARTE CORRESPONDIENTE A LA RECEPCIÓN B *
  // Siempre que haya información disponible vía RF...
  // **** PARTE CORRESPONDIENTE A LA RECEPCIÓN A ***************************

     // radio.startListening(); // Comienzo a escuchar por el canal "1"

      // Siempre que haya información disponible vía RF...
      while(radio.available())
      {
        // Se recibe el valor de la variable estado_pulsador_B[0] a través de RF
        radio.read(PwmSig, sizeof(PwmSig));
       uno=PwmSig[0];
       Serial.println(PwmSig[0]);


      }
      delay(20); // Doy tiempo de lectura al receptor A


      analogWrite(pwm, uno);

  delay(1000);
  }
```

Anexo 5: Códigos para la programación del variador

6 Descripción general de parámetros

Descripción general de los parámetros

0-** Func. / Display
0-0* Ajustes básicos
0-03 Ajustes regionales
[0] Internacional
*[1] US
0-04 Estado de func. al conectar (manual)
[0] Auto-arranque
*[1] Par. forz., ref. guard
[2] Par. forz., ref. = 0
0-1* Gestión de ajustes
0-10 Ajuste activo
*[1] Ajuste 1
[2] Ajuste 2
[9] Ajuste múltiple
0-11 Editar ajuste
*[1] Ajuste 1
[2] Ajuste 2
[9] Ajuste activo
0-12 Ajustes enlazados
[0] Sin enlazar
*[20] Enlazado
0-31 Valor mín. de lectura def. por usuario
0,00 – 9999,00
*0,00
0-32 Valor máx. de lectura defin. usuario
0,00 – 9999,00
*100,0
0-4* Teclado
0-40 Botón [Manual] en Teclado
[0] Desactivado
*[1] Activado
0-41 Botón [Off / Reset] en Teclado
[0] Desactivar todos
*[1] Activar todos
[2] Activar solo Reset
0-42 Botón [Auto] en Teclado
[0] Desactivado
*[1] Activado
0-5* Copiar/Guardar
0-50 Teclado Copiar
*[0] No copiar
[1] Trans Teclado tod. par.
[2] Tr d Teclado tod. par.
[3] Tr d Teclado par ind tam
0-51 Copia de ajuste
*[0] No copiar
[1] Copiar del ajuste 1
[2] Copiar del ajuste 2
[9] Copiar del ajuste de fábrica
0-6* Contraseña
0-60 Contraseña menú principal
0 - 999 * 0
1-** Carga/motor
1-0* Ajustes generales
1-00 Modo de configuración
*[0] Veloc. Lazo Abierto
[3] Proceso
1-01 Principio Control Motor
[0] U/f
*[1] Control Vector Adv.
1-03 Características de par
*[0] Par constante
[2] Ahorro de energía
1-05 Configuración modo local
[0] Lazo Abierto de velocidad
*[2] Según par. 1-00
1-2* Datos de motor
1-20 Potencia motor [kW] [CV]
[1] 0,09 kW/0,12 CV
[2] 0,12 kW/0,16 CV
[3] 0,18 kW/0,25 CV
[4] 0,25 kW/0,33 CV
[5] 0,37 kW/0,50 CV
[6] 0,55 kW/0,75 CV
[7] 0,75 kW/1,00 CV
[8] 1,10 kW/1,50 CV
[9] 1,50 kW/2,00 CV
[10] 2,20 kW/3,00 CV
[11] 3,00 kW/4,00 CV
[12] 3,70 kW/5,00 CV
[13] 4,00 kW/5,40 CV
[14] 5,50 kW/7,50 CV
[15] 7,50 kW/10,00 CV
[16] 11,00 kW/15,00 CV
1-22 Tensión motor
50 - 999 V * 230 - 400
1-23 Frecuencia motor
20 - 400 Hz * 60 Hz
1-24 Intensidad motor
0,01 - 26,00 A * Dep. tipo motor
1-25 Veloc. nominal motor
100 - 9.999 rpm * Dep. tipo motor
1-29 Autoajuste
*[0] Off
[2] Activar Autoajuste
1-3* Dat. avanz. motor
1-30 Resistencia estátor (Rs)
[Ohm] * Dependiente de los datos del motor
1-33 Reactancia fuga estátor (X1)
[Ohm] * Dep. de datos del motor
1-35 Reactancia princ. (Xh)
[Ohm] * Dep. on motor data
1-5* Indep. de carga de la carga
1-50 Magnet. motor a veloc. cero
0 - 300 % * 100 %
1-52 Velocidad mínima a magn. normal [Hz]
0,0 - 10,0 Hz * 0,0 Hz
1-55 Característica U/f - U
0 - 999,9 V
1-56 Característica U/f - F
0 - 400 Hz
1-6* Aj. depend. de la carga
1-60 Velocidad baja Compensación de carga
0 - 199 % * 100 %
1-61 Compensación carga alta velocidad
0 - 199 % * 100 %
1-62 Compensación de deslizamiento
-400 - 399 % * 100 %
1-63 Constante de tiempo compens. deslizam.
0,05 - 5,00 s * 0,10 s
1-7* Ajustes arranque
1-71 Mant. tiempo
0,0 - 10,0 s * 0,0 s
1-72 Función de arranque
[0] CC mant./tiempo ret.
[1] Freno CC/tiempo retar.
*[2] Inerc. / tiempo retardo
1-73 Modo arranque
*[0] Desactivado
[1] Activar - Enganche Carga en Giro
1-8* Ajustes de parada
1-80 Función de parada
*[0] Inercia
[1] CC mantenida
1-82 Vel. mín. para func. parada [Hz]
0,0 - 20,0 Hz * 0,0 Hz
1-9* Temperatura del motor
1-90 Protección térmica del motor
*[0] Sin protección
[1] Advert. termistor
[2] Termistor desconexión
[3] Advertencia de sobrecarga electrónica
[4] Desconexión por sobrecarga electrónica
1-93 Fuente de termistor
*[0] Ninguno
[1] Entrada analógica 53
[6] Entrada digital 29
2-** Frenos
2-0* Freno de CC
2-00 Intensidad de CC mantenida
0 - 150 % * 50 %
2-01 Intens. freno CC
0 - 150 % * 50 %
2-02 Tiempo de frenado CC
0,0 - 60,0 s * 10,0 s
2-04 Velocidad de conexión del freno CC
0,0 - 400,0 Hz * 0,0 Hz
2-1* Func. energ. freno
2-10 Función de freno
*[0] Off
[1] Freno con resistencia
[2] Frenado de CA
2-11 Resistencia de freno (ohm)
5 - 5.000 * 5
2-16 Intensidad máx. de frenado de CA
0 - 150 % * 100 %
2-17 Control de sobretensión
*[0] Desactivado
[1] Activado (no parada)
[2] Activado
2-2* Freno mecánico
2-20 Intensidad freno liber.
0,00 - 100,0 A * 0,00 A
2-22 Activar velocidad freno [Hz]
0,0 - 400,0 Hz * 0,0 Hz
3-** Ref./Rampas
3-0* Límites referencia
3-00 Rango de referencia
*[0] Mín - Máx
[1] -Máx - +Máx
3-02 Referencia mínima
-4.999 - 4.999 * 0,000
3-03 Referencia máxima
-4.999 - 4.999 * 50,00
3-1* Referencias
3-10 Referencia interna
-100,0 - 100,0 % * 0,00 %
3-11 Velocidad fija [Hz]
0,0 - 400,0 Hz * 5,0 Hz
3-12 Valor de enganche/arriba-abajo
0,00 - 100,0 % * 0,00 %

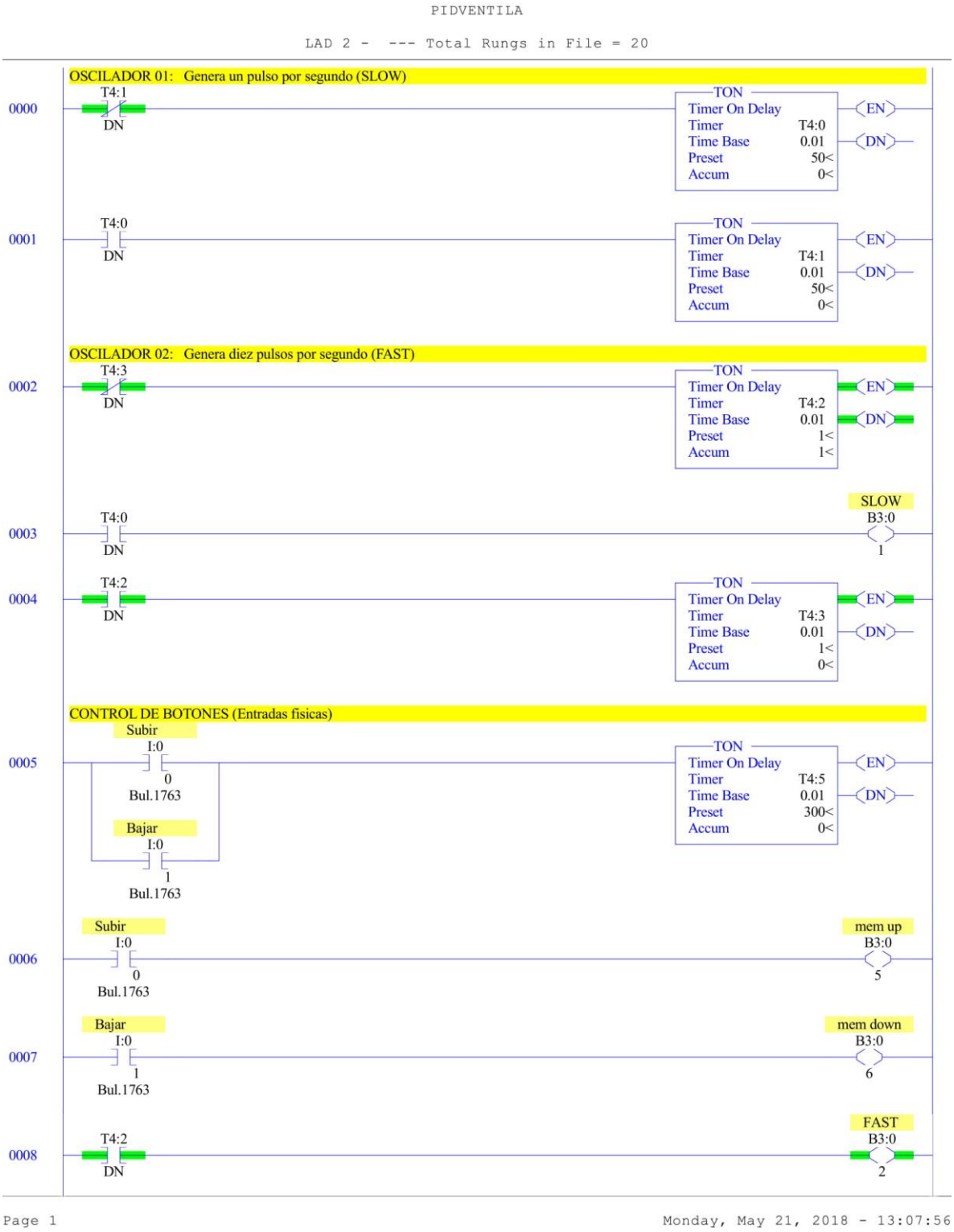
3-14 Referencia relativa interna
-100,0 - 100,0 % * 0,00 %
3-15 Recurso de referencia 1
[0] Sin función
*[1] Entrada analógica 53
[2] Entrada analógica 60
[8] Entrada de pulsos 33
[11] Ref. bus local
[21] Potenciómetro Teclado
3-16 Recurso de referencia 2
[0] Sin función
[1] Entrada analógica 53
*[2] Entrada analógica 60
[8] Entrada de pulsos 33
[11] Ref. bus local
[21] Potenciómetro Teclado
3-17 Recurso de referencia 3
[0] Sin función
[1] Entrada analógica 53
[2] Entrada analógica 60
[8] Entrada de pulsos 33
*[11] Ref. bus local
[21] Potenciómetro Teclado
3-18 Recurso escal. rel. de referencia
*[0] Sin función
[1] Entrada analógica 53
[2] Entrada analógica 60
[8] Entrada de pulsos 33
[11] Ref. bus local
[21] Potenciómetro Teclado
3-4* Acel/Decel 1
3-40 Patrón Acel/Decel 1
*[0] Lineal
[2] Forma en S
3-41 Intervalo tiempo acel. 1
0,05 - 3.600 s * 3,00 s
3-42 Intervalo desacel. 1
0,05 - 3.600 s * 3,00 s
3-5* Acel/Desacel. 2
3-50 Patrón Acel/Decel 2
*[0] Lineal
[2] Forma en S
3-51 Tiempo acel. 2
0,05 - 3.600 s * 3,00 s
3-52 Intervalo desacel. 2
0,05 - 3.600 s * 3,00 s
3-8* Otras rampas

3-80 Intervalo de acel. y desacel. lento
0,05 - 3.600 s * 3,00 s
3-81 Parada rápida Desacel. Time
0,05 - 3.600 s * 3,00 s
4-** Lím./Advert.
4-1* Límites motor
4-10 Bloqueo inverso
[0] Bloqueo inverso
[1] Inverso
*[2] Ambos sentidos
4-12 Límite bajo veloc. motor [Hz]
0,0 - 400,0 Hz * 0,0 Hz
4-14 Límite alto veloc. motor [Hz]
0,1 - 400,0 Hz * 65,0 Hz
4-16 Modo motor límite de par
0 - 400 % * 150 %
4-17 Modo generador límite de par
0 - 400 % * 100 %
4-5* Ajuste advertencias
4-50 Advert. Intens. baja
0,00 - 26,00 A * 0,00 A
4-51 Advert. Intens. alta
0,00 - 26,00 A * 26,00 A
4-58 Función Fallo fase motor
[0] Off
*[1] On
4-6* Salto de frecuencias
4-61 Salto de frecuencias De [Hz]
0,0 - 400,0 Hz * 0,0 Hz
4-63 Salto de frecuencia [Hz]
0,0 - 400,0 Hz * 0,0 Hz
5-1* Entradas digitales
5-10 Terminal 18 entrada digital
[0] Sin función
[1] Reinicio
[2] Inercia
[3] Inercia y reinicio
[4] Parada rápida
[5] Freno CC inv.
[6] Parada
*[8] Arranque
[9] Arranque por pulsos
[10] Cambio de sentido
[11] Arranque e inversión
[12] Act. arranque adelan.
[13] Act. arranque inverso
[14] Veloc. fija
[16-18] Ref.interna EXB
[19] Mantener referencia
[20] Mant. salida
[21] Aceleración
[22] Deceleración
[23] Selec.ajuste LSB
[28] Engan. arriba
[29] Enganc. abajo
[34] Bit 0 rampa
[60] Contador A (ascend.)
[61] Contador A (descend.)
[62] Reset contador A
[63] Contador B (ascend.)
[64] Contador B (descend.)
[65] Reset del contador B
5-11 Terminal 19 entrada digital
Vea el par. 5-10. * [10] Cambio de sentido
5-12 Terminal 27 entrada digital
Vea el par. 5-10. * [1] Reinicio
5-13 Terminal 29 entrada digital
Vea el par. 5-10. * [14] Veloc. fija
5-15 Terminal 33 entrada digital
Vea el par. 5-10. [16] Ref.interna LSB
[26] Parada precisa
[27] Arranq./parada prec.
[32] Entra de pulsos
5-4* Relés
5-40 Relé de función
*[0] Sin función
[1] Ctrl prep.
[2] Conv. preparado
[3] Conv. preparado, remoto
[4] Activar / sin advert.
[5] Unidad en func.
[6] Func./sin advert.
[7] Func. en ran./sin adv.
[8] Func. en ref./sin adv.
[9] Alarma
[10] Alarma o advertencia
[12] Fuera ran. intensidad
[13] Corriente posterior, baja
[14] Corriente anterior, alta
[21] Advertencia térmica
[22] Listo, sin adv. térm.
[23] Rem list sin adv tér
[24] Listo, tensión OK
[25] Cambio sentido
[26] Bus OK
[28] Freno, sin advert.
[29] Fren. prep. sin fallos
[30] Fallo freno (IGBT)
[32] Ctrl. freno mec.
[36] Bit cód. control 11
[51] Ref. local activa
[52] Ref. remota activa
[53] Sin alarma
[54] Coman. arran. activo
[55] Func. inverso
[56] Convertidor de frecuencia en modo manual
[57] Conv. en modo auto.
[60-63] Comparador 0-3
[72-73] Regla lógica 03
[81] Salida digital B del controlador lógico
5-5* Entrada de pulsos
5-55 Term. 33 baja frecuencia
20 - 4.999 Hz * 20 Hz
5-56 Term. 33 alta frecuencia
21 - 5.000 Hz * 5.000 Hz
5-57 Term. 33 valor bajo ref. /realim
-4.999 - 4.999 * 0,000
5-58 Term. 33 valor alto ref. /realim
-4.999 - 4.999 * 50.000
6-** E/S analógica
6-0* Modo E/S analógico
6-00 Tiempo Límite Cero Activo
1 - 99 s * 10 s
6-01 Tiempo Límite Cero Activo
*[0] Off
[1] Mant. salida
[2] Parada
[3] Velocidad fija
[4] Velocidad máx.
[5] Parada y desconexión
6-1* Entrada analógica 1
6-10 Tensión baja Terminal 53
0,00 - 9,99 V * 0,07 V
6-21 Tensión alta Terminal 53
0,01 - 10,00 V * 10,00 V
6-22 Terminal 53 Intensidad baja
0,00 - 19,99 mA * 0,14 mA

6-23 Intensidad alta terminal 53
0,01 - 20,00 mA * 20,00 mA
6-14 Term. 53 valor bajo ref. /realim
-4,999 - 4,999 * 0,000
6-15 Term. 53 valor alto ref. /realim
-4,999 - 4,999 * 50,000
6-16 Terminal 54 constante tiempo filtro
0,01 - 10,00 s * 0,01 s
6-19 Terminal 53 modo
*[0] Modo V
[1] Modo mA
***6-2* Entrada analógica 2**
6-22 Terminal 60 escala baja mA
0,00 - 19,99 mA * 0,14 mA
6-23 Terminal 60 escala alta mA
0,01 - 20,00 mA * 20,00 mA
6-24 Term. 60 valor bajo ref. /realim
-4,999 - 4,999 * 0,000
6-25 Term. 60 valor alto ref. /realim
-4,999 - 4,999 * 50,00
6-26 Terminal 60 constante tiempo filtro
0,01 - 10,00 s * 0,01 s
***6-8* Potenciómetro Teclado**
6-81 Teclado potenciómetro Referencia baja
-4,999 - 4,999 * 0,000
6-82 Teclado potenciómetro Referencia alta
-4,999 - 4,999 * 50,00
6-9* Salida analógica *xx
6-90 Modo terminal 42
*[0] 0-20 mA
[1] 4-20 mA
[2] Salida digital
6-91 Terminal 42 salida analógica
*[0] Sin función
[10] Frecuencia de salida
[11] Referencia
[12] Realimentación
[13] Intensidad motor
[16] Potencia
[20] Control de bus
6-92 Terminal 42 salida digital
Consulte par. 5-40.
* [0] Sin función
[1] Mant. salida
[2] Parada
[3] Velocidad fija
[4] Vel. máx.
[5] Parada y desconexión
8-06 Reiniciar si tiempo límite cód. ctrl.
*[0] Sin función
[1] Reiniciar
***8-3* Convertidor GE Configuración de puerto**
8-30 Protocolo
*[0] Convertidor GE
[2] Modbus RTU
8-31 Dirección
1 - 247 * 1
8-32 Convertidor GE Velocidad del puerto en baudios
[0] 2.400 baudios
[1] 4.800 baudios
*[2] 9.600 baudios
[3] 19.200 baudios
*[4] 38.400 baudios
8-33 Convertidor GE Paridad de puerto
*[0] Paridad par, 1 bit parada
[1] Paridad impar, 1 bit parada
[2] Sin paridad, 1 bit parada
[3] Sin paridad, 2 bits parada
8-35 Retardo respuesta mín.
0,001-0,5 * 0,010 s
8-36 Retardo respuesta máx.
0,100 - 10,00 s * 5,000 s
***8-5* Digital/Bus**
8-50 Selección inercia
[0] Entrada digital
[1] Bus
[2] Lógico Y
*[3] O Lógico
8-51 Selección parada rápida
Vea el par. 8-50 * [3] O Lógico
8-52 Selección freno CC
Vea el par. 8-50 * [3] O Lógico
8-53 Selec. arranque
Vea el par. 8-50 * [3] O Lógico
8-54 Selec. sentido inverso
Vea el par. 8-50 * [3] O Lógico
8-55 Selec. ajuste
Vea el par. 8-50 * [3] O Lógico
8-56 Selec. referencia interna
Vea el par. 8-50 * [3] O Lógico
***8-9* Vel. fija del bus / Realimentación**
8-94 Realim. de bus 1
0x8000 - 0x7FFF * 0
13-** Control lógico
13-0* Control lógico SLC
13-00 Modo Control Lógico
*[0] Off
[1] On
13-01 Evento arranque
[0] Falso
[1] Verdadero
[2] En funcionamiento
[3] En rango
[4] En referencia
[7] Fuera ran. intensidad
[8] I posterior bajo
[9] I anterior alto
[16] Advertencia térmica
[17] Tens. alim. fuera ran.
[18] Cambio de sentido
[19] Advertencia
[20] Descon._alarma
[21] Bloq._descon._alarma
[22-25] Comparador 0-3
[26-29] Regla lógica 0-3
[33] EntradaDigital_18
[34] EntradaDigital_19
[35] EntradaDigital_27
[36] EntradaDigital_29
[38] EntradaDigital_33
*[39] Comando de arranque
[40] Convert. parado
13-02 Evento parada
Véase el par. 13-41 * [40] Convertidor parado
13-03 Reset SLC
*[0] No reiniciar
[1] Reset SLC
13-1* Comparadores

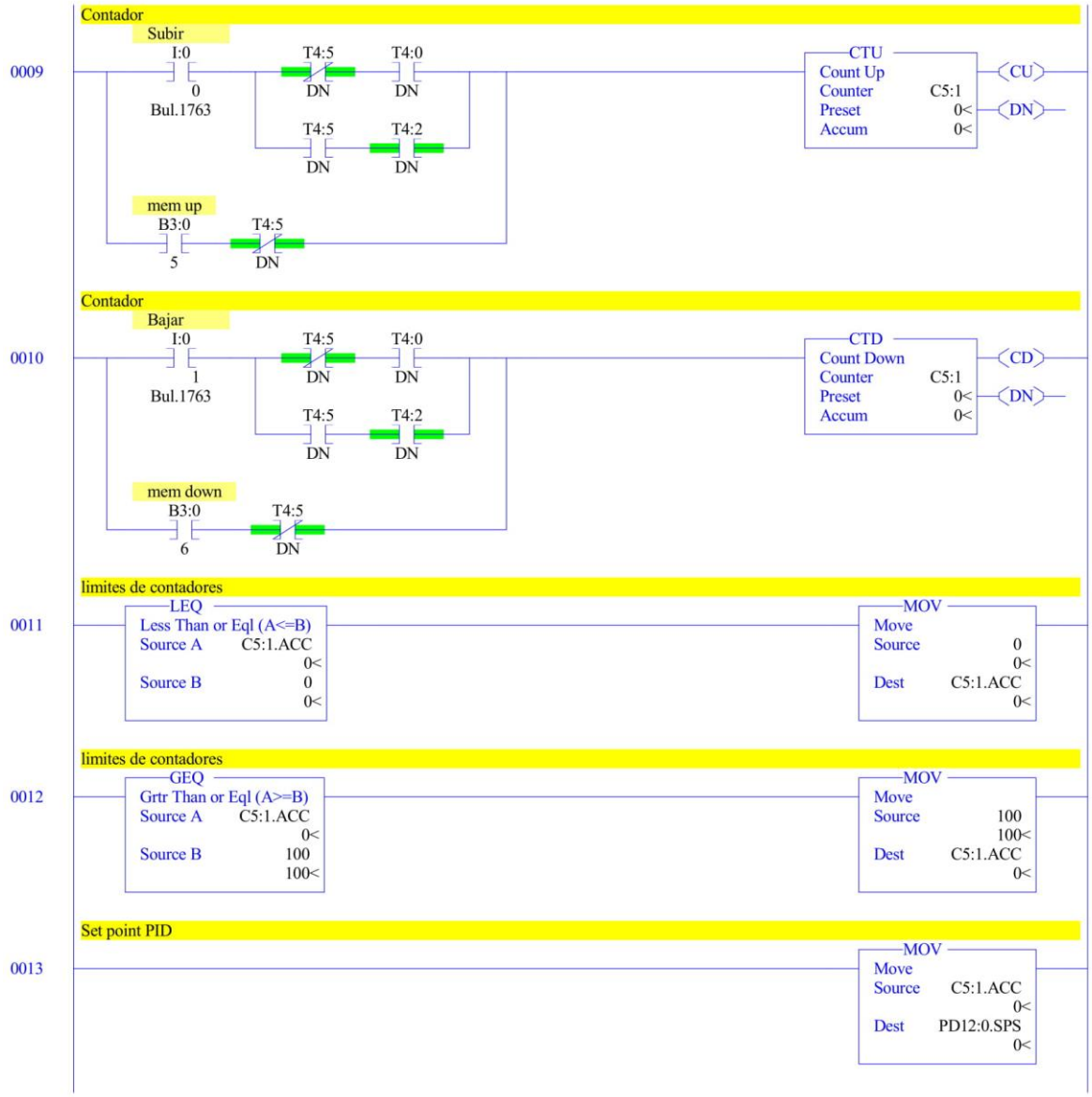
13-10 Operando comparador
*[0] Desactivado
[1] Referencia
[2] Realimentación
[3] Veloc. motor
[4] Intensidad motor
[6] Potencia motor
[7] Tensión motor
[8] Tensión Bus CC
[12] Entrada analógica 53
[13] EntradaAnalógica 60
[18] Entrada de pulsos 33
[20] Número de alarma
[30] Contador A
[31] Contador B
13-11 Operador comparador
[0] Menor que
*[1] Aprox. igual
[2] Mayor que
13-12 Valor comparador
-9.999 - 9.999 * 0,0
13-2* Temporizadores
13-20 SL Temporizador del controlador
0,0 - 3.600 s * 0,0 s
13-4* Reglas lógicas
13-40 Regla lógica boleana 1
Vea el par. 13-01 * [0] Falso
[30] - [32] SL Tiempo límite de desconexión 0-2
13-41 Operador regla lógica 1
*[0] Desactivado
[1] And
[2] Or
[3] And not
[4] Or not
[5] Not and
[6] Not or
[7] Not and not
[8] Not or not
13-42 Regla lógica boleana 2
Consulte par. 13-40
13-43 Operador regla lógica 2
Vea el par. 13-41 * [0] Desactivado
13-44 Regla lógica boleana 3
Consulte par. 13-40
13-5* Estados
13-51 SL Evento del controlador
Consulte par. 13-40
13-52 SL Acción del controlador
*[0] Desactivado
[1] Sin acción
[2] Selección de ajuste 1
[3] Selección de ajuste 2
[10-17] Selec. ref. presel. 0-7
[18] Seleccione acel./desacel. 1.
[19] Seleccione acel./desacel. 2.
[22] En funcionamiento
[23] Func. sentido inverso
[24] Parada
[25] Parada rápida
[26] DCstop
[27] Inercia
[28] Mant. salida
[29] Tempor. inicio 0
[30] Tempor. inicio 1
[31] Tempor. inicio 2
[32] Aj. sal. dig. A baja
[33] Aj. sal. dig. B baja
[38] Aj. sal. dig. A alta
[39] Aj. sal. dig. B alta
[60] Reset del contador A
[61] Reset del contador B
14-** Func. especiales
14-0* Frecuencia portadora
14-01 Ruido del motor (frecuencia portadora variable)
[0] 2 kHz
*[1] 4 kHz
[2] 8 kHz
[4] 16 kHz
14-03 Sobremodulación
[0] Off
*[1] On
14-1* Control alimentación
14-12 Función desequil. alimentación
*[0] Desconexión
[1] Advertencia
[2] Desactivado
14-2* Reinicio desconex.
14-20 Modo Reset
*[0] Reset manual
[1-9] Reset autom. 1-9
[10] Reset autom. 10
[11] Reset autom. 15
[12] Reset autom. 20
[13] Reinic. auto. infinito
14-21 Tiempo de reinicio automático
0 - 600 s * 10 s
14-22 Restaurar ajustes de fábrica
*[0] Funcionam. normal
[2] Restaurar ajustes de fábrica
14-26 Acción en Fallo del convertidor de frecuencia
*[0] Desconexión
[1] Advertencia
14-4* Optimización de Ahorro de energía
14-41 Ahorro de energía Magnetización mínima
40 - 75 % * 66 %
15-** Información convertidor
15-0* Datos func.
15-00 Días de funcionamiento
15-01 Horas funcionam.
15-02 Contador kWh
15-03 Arranques
15-04 Sobretemperat.
15-05 Sobretensión
15-06 Reiniciar contador KWh
*[0] No reiniciar
[1] Reiniciar contador
15-07 Reinicio contador de horas funcionam.
*[0] No reiniciar
[1] Reiniciar contador
15-3* Registro fallos
15-30 Registro fallos: Código de fallo
15-4* Id. convertidor
15-40 Tipo de convertidor GE
15-41 Sección de potencia
15-42 Tensión
15-43 Versión de software
15-46 Nº de pedido convertidor de frecuencia
15-48 Teclado Nº ID
15-51 Nº serie convert. frecuencia
16-** Lecturas de datos
16-0* Estado general
16-00 Código de control
0 - 0XFFFF
16-01 Referencia [Unidad]
-4999 - 4999
16-02 Referencia %
-200,0 - 200,0 %
16-03 Cód. estado
0 - 0XFFFF
16-05 Valor real princ. [%]
-200,0 - 200,0 %
16-09 Lectura personalizada
Consulte además los par. 0-31, 0-32 y 4-14.
16-1* Estado motor
16-10 Potencia [kW]
16-11 Potencia [CV]
16-12 Tensión del motor [V]
16-13 Frecuencia [Hz]
16-14 Intensidad motor [A]
16-15 Frecuencia [%]
16-18 Térmico motor [%]
16-3* Estado convertidor
16-30 Tensión Bus CC
16-34 Temp. disipador
16-35 Térmico inversor
16-36 Corriente nominal del convertidor
16-37 Intensidad máxima convertidor
16-38 Estado del controlador lógico
16-5* Ref. y realim.
16-50 Referencia externa
16-51 Referencia de pulsos
16-52 Realimentación [Unit]
16-6* Entradas y salidas
16-60 Entrada digital 18,19,27,33
0 - 1111
16-61 Entrada digital 29
0 - 1
16-62 Entrada analógica 53 (tensión)
16-63 Entrada analógica 53 (intensidad)
16-64 Entrada analógica 60
16-65 Salida analógica 42 [mA]
16-68 Ent. pulsos [Hz]
16-71 Salida Relé [bin]
16-72 Contador A
16-73 Contador B
16-8* Bus de campo / Convertidor GE Puerto
16-86 Convertidor GE Puerto REF 1
0x8000 - 0x7FFF
16-9* Lect. diagnóstico
16-90 Código de alarma
0 - 0XFFFFFFFF
16-92 Cód. de advertencia
0 - 0XFFFFFFFF
16-94 Cód. ampliado
0 - 0XFFFFFFFF

Anexo 6: Programa Control PID para PLC



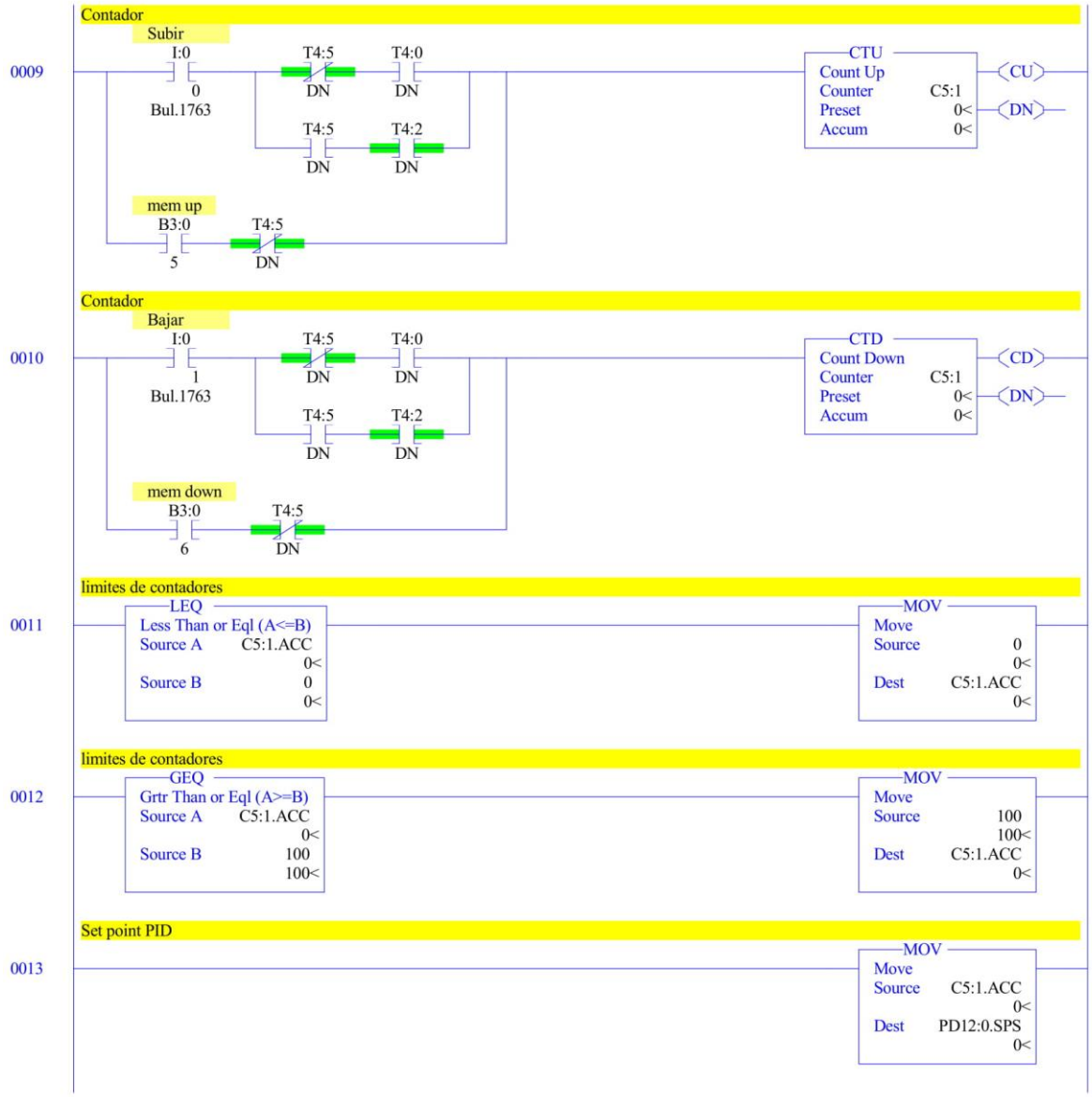
PIDVENTILA

LAD 2 - --- Total Rungs in File = 20



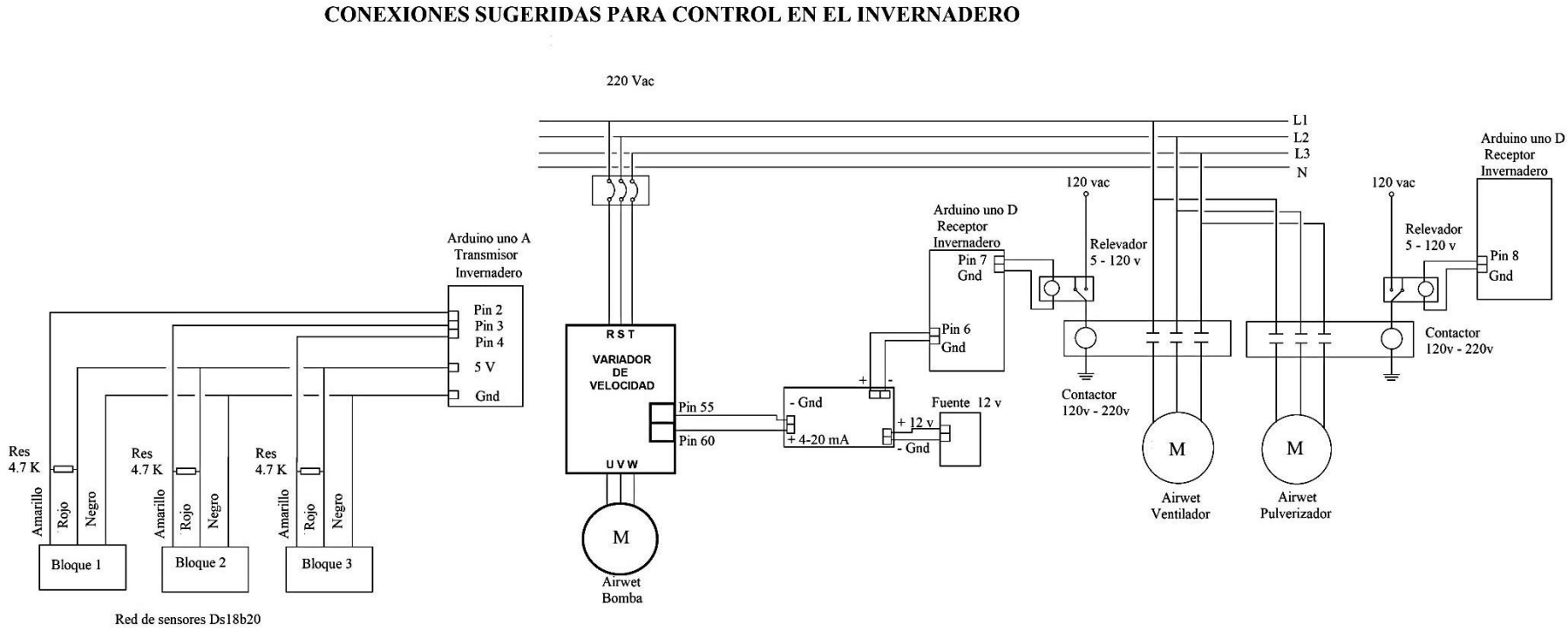
PIDVENTILA

LAD 2 - --- Total Rungs in File = 20



Anexo 7: Diagrama de conexión sugerido para control neuronal

En este diagrama se muestra la salida a un solo variador, para el completo control se deberán utilizar tres Airwet mas, realizando las conexiones y ajustes de programación pertinentes.



Anexo 8: Imágenes de pruebas de campo

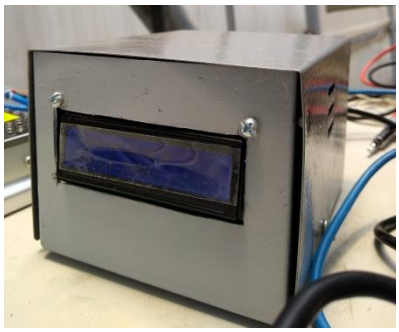


Figura Transmisor/receptor en invernadero

Figura Prueba de alcance de aspersión del Airwet

Figura Variador de velocidad funcionando

Figura Humificador en funcionamiento

Figura Placa de acoplamiento de sensores

Anexo 9: Diagrama de bloques de control RNA

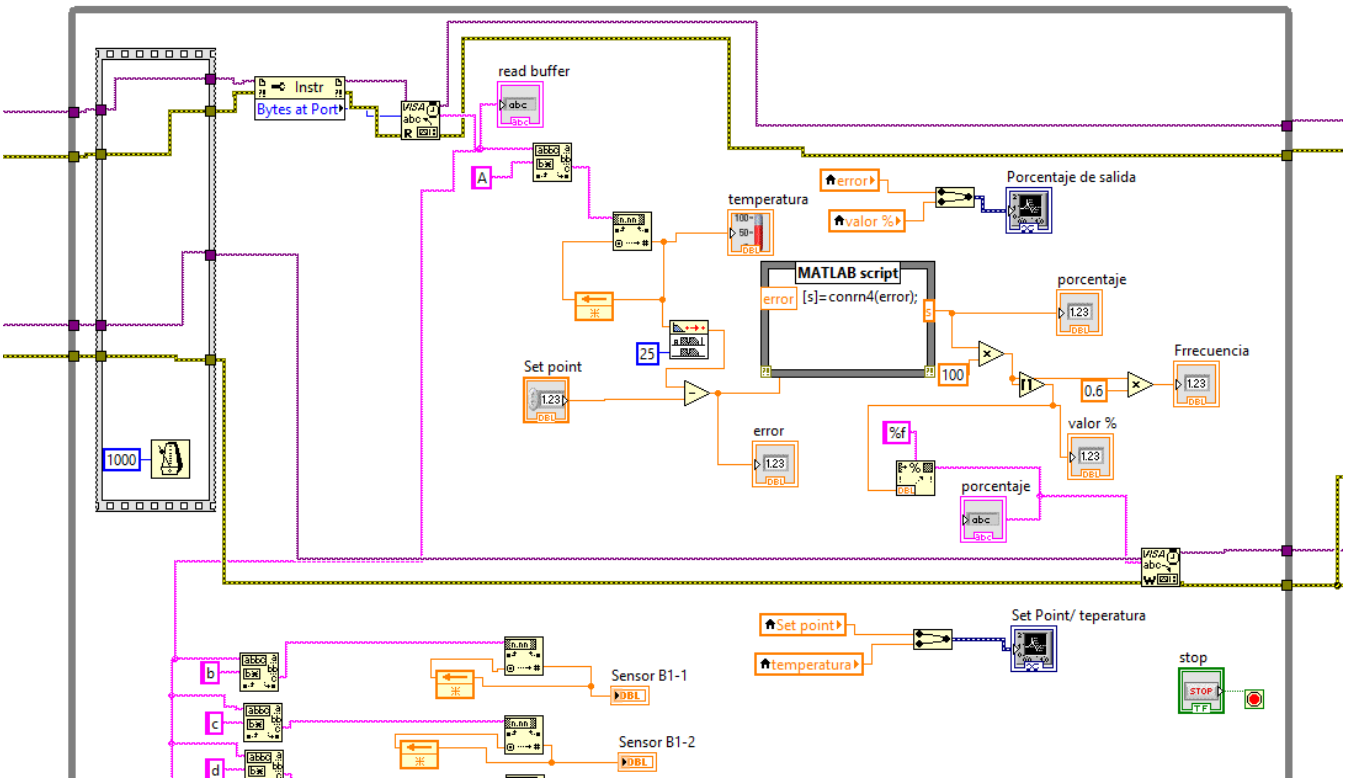


Figura Diagrama de bloques control RNA

Anexo 10: Data sheet Arduino uno

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-9V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) (0.5 KB used by bootloader)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

2 Arduino Uno

Schematic & Reference Design

EAGLE files: arduino-uno-reference-design.zip

Schematic: arduino-uno-schematic.pdf

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm centre-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

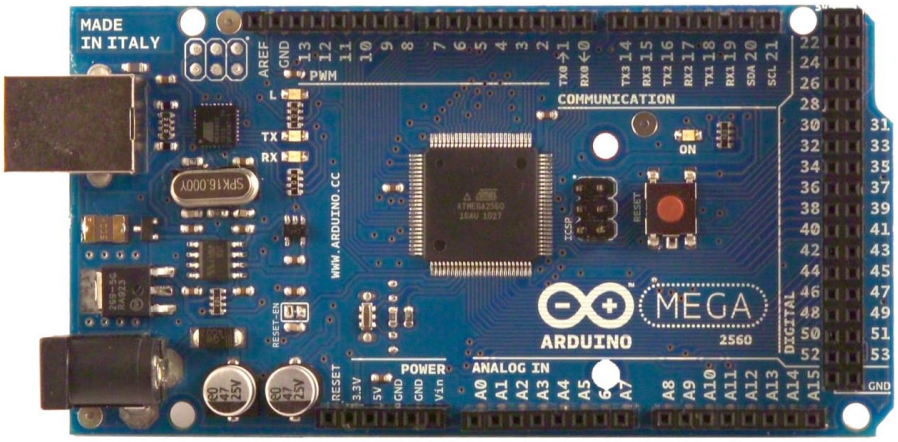
Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

Anexo 11: Data sheet Arduino Mega

Arduino MEGA 2560



CE

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

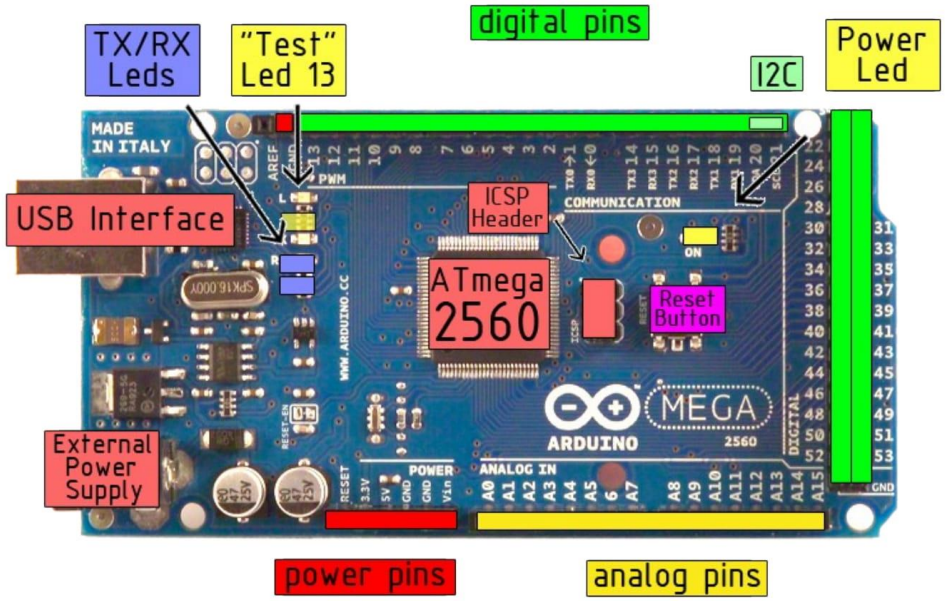
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Anexo 12: Como referenciar la bibliografía consultada

a) **Libros;** Autor. (Año). Título subrayado. Editorial. Lugar de impresión. Número de las páginas consultadas. Ejemplo:

[01] Ruelas-Lepe, Rubén y Bernal-Casillas, José de Jesús. (2010). Desarrollo y publicación de la tesis. Amate editorial. Zapopan, Jalisco, México. Páginas 98-112.

b) **Artículos;** Autor. (Año). Título. Nombre del artículo entre comillas. Subrayar el evento en el cual se presentó el artículo. Tomo, número o cualquier otra indicación que contenga la portada de la revista, tal y como se indica en la mismo.

[02] García-Cortés, José de Jesús y otros. (2017). "Modelado y control de la variable temperatura de un invernadero por medio de un controlador PID clásico y un controlador PID difuso de 9 reglas". Academia Journals Cd. Juárez 2016. Volumen II; Artículo No. J179; ISSN 1946-5351. Cd. Juárez, Chihuahua, México.

c) **Internet;** Autor. (año). Título subrayado. Dirección. Fecha de consulta. Ejemplo:

[03] Juma, N. G. (1999). The Pedosphere and its Dynamics: Ecological Functions of Soil, 2.4 Recycles Wastes. En línea. Disponible en: www.pedosphere.com. Consultada en diciembre de 1999.

d) **Conferencias o clases;** Autor. (Año). Indicar si es conferencia o clase. "tema". Materia especifica subrayada. Lugar, fecha. Ejemplo:

[04] Molina-Gaudo, Pilar. (2005). Conferencia. "La mujer en la ingeniería". Educación superior. Universidad de Zaragoza; Zaragoza, España. 14 de abril.

e) **Experiencia;** Autor. Experiencia. Lugar donde se adquirió la experiencia. Materia impartida o proyecto realizado. Periodo. Ejemplo:

[05] Pérez-Quiroz, Raúl. Experiencia. Instituto Tecnológico de San Luis Potosí. Asignatura de programación de enero a junio de 1986