ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DESARROLLO E IMPLEMENTACION DE LIBRERIAS DE CONTROL INDUSTRIAL BASADAS EN EL SISTEMA OPERATIVO FreeRTOS, PARA EL CONTROLADOR LOGICO PROGREMABLE CRONOS

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y AUTOMATIZACIÓN

RAFAEL SIAMAK VACA NOBOA

rafael.vaca@epn.edu.ec

DIRECTOR: NELSON GONZALO SOTOMAYOR OROZCO, MSc.

nelson.sotomayor@epn.edu.ec

Quito, febrero 2022

CERTIFICACIONES

Yo, Rafael Siamak Vaca Noboa declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

Rafael Siamak Vaca Noboa

Certifico que el presente trabajo de integración curricular fue desarrollado por Rafael Siamak Vaca Noboa, bajo mi supervisión.

Firmado digitalmente por NELSON GONZALO SOTOMAYOR OROZCO - 1710436724
Fecha: 2022.02.14 12:12:39 -05'00'

Nelson Gonzalo Sotomayor Orozco, MSc.

DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

Rafael Siamak Vaca Noboa

Nelson Gonzalo Sotomayor Orozco, MSc.

DEDICATORIA

El presente trabajo de titulación está dedicado a mi madre, Cecilia Noboa, por ser un pilar fundamental en mi desarrollo personal y por haber sido un apoyo a lo largo de toda mi vida, inculcándome valores y a seguir adelante ante cualquier adversidad.

ÍNDICE DE CONTENIDO

RESUMEN

En este trabajo se presenta el diseño de librerías en la interfaz de Arduino para elaboración de bloques de función predefinidos los cuales están basados en la programación de controladores lógicos programables.

Se implementa un sistema operativo (FreeRTOS) en el controlador lógico programable Cronos, para lo cual se incluye una librería en la plataforma de Arduino, este sistema operativo permitirá al microcontrolador trabajar de manera paralela, dividiendo los procesos en tareas para poder trabajar con ellas de manera más optima ya que se les dará orden de prioridad y tendrá un mejor manejo de los tiempos de ejecución.

Se realiza un proceso automático incluyendo el sistema operativo y las librerías de bloques de función predefinidas diseñadas, las cuales van a trabajar en dicho proceso con la finalidad de comprobar el funcionamiento del sistema operativo. El PLC será sometido a pruebas en un laboratorio especializado con el objetivo de garantizar su funcionamiento en la industria.

PALABRAS CLAVE: Bloques de función, controlador lógico programable, Arduino, FreeRTOS.

ABSTRACT

This work presents the design of libraries in the Arduino interface for the elaboration of predefined function blocks which are based on the programming of programmable logic controllers.

An operating system (FreeRTOS) is implemented in the Cronos programmable logic controller, for which a library is included in the Arduino platform, this operating system will allow the microcontroller to work in parallel, dividing the processes into tasks to be able to work with them in a parallel way. more optimal since they will be given priority order and will have a better management of execution times.

An automatic operating process is carried out including the system and the predefined function block libraries which will work in said process in order to check the operation of the operating system. The PLC will be tested in a specialized laboratory in order to guarantee its operation in the industry.

KEYWORDS: Function blocks, programmable logic controller, Arduino, FreeRTOS.

1 INTRODUCCION

El sector industrial ecuatoriano, para pequeñas empresas, se encuentra en constante crecimiento lo que ha producido un incremento en la demanda de controladores lógicos programables, ya que es un factor importante para los procesos industriales. Por esto se ha partido de un controlador lógico programable desarrollado por la Escuela Politécnica Nacional que fue diseñado a partir de un microcontrolador (STM32F103), al que se le implementa un sistema operativo FreeRTOS, que le permite trabajar de manera casi similar a un PLC comercial, teniendo una distribución de tareas en modo estático. Con el fin de facilitar la programación del PLC mencionado se implementa 4 librerías de bloques de función predefinidos, con la finalidad de reducir las líneas de programación y permitir a los usuarios familiarizarse con la programación en Arduino. Con esto se pretende dar a los usuarios una programación más sencilla y accesible para sus controladores lógicos programables, haciendo que cualquier persona con un poco de criterio de programación, pueda manipular los códigos y realizar sus propios procesos industriales.

1.1 OBJETIVO GENERAL

Desarrollar e implementar de librerías de control industrial basadas en el sistema operativo FreeRTOS, para el controlador lógico programable CRONOS.

1.2 OBJETIVOS ESPECÍFICOS

1. Realizar una síntesis sobre el sistema operativo FreeRTOS y su aplicación a microcontroladores, sobre el PLC Cronos y sobre los 4 bloques de control industrial más utilizados.
2. Desarrollar un mínimo de 4 librerías de control industrial para el PLC CRONOS basadas en el sistema operativo FreeRTOS.
3. Implementar el sistema operativo FreeRTOS y las librerías en el PLC CRONOS.
4. Realizar la programación de 1 proceso automático, en los cuales se aplican las librerías desarrolladas para el PLC CRONOS.
5. Implementar los procesos en un laboratorio especializado, para verificar el funcionamiento de las librerías en el PLC CRONOS.

1.3 ALCANCE

- Se realizará una revisión bibliográfica sobre el sistema operativo FreeRTOS, enfocándose en manejo de tareas estáticas y dinámicas, aplicaciones a microcontroladores y aplicaciones industriales.
- Se realizará una revisión bibliográfica sobre el PLC Cronos enfocándose en sus principales características.
- Se realizará una revisión bibliográfica sobre los 4 bloques de control industrial más utilizados.
- Se desarrollará un mínimo de 4 librerías enfocadas a procesos industriales, que puedan trabajar en base al sistema operativo FreeRTOS. Las librerías serán desarrolladas en Arduino IDE con el objetivo de trabajar con el PLC CRONOS.
- Se instalará el sistema operativo FreeRTOS en el PLC CRONOS y se implementarán las librerías desarrolladas.
- Se desarrollará la programación de 2 procesos automáticos, los cuales serán implementados en el PLC cronos utilizando las librerías desarrolladas.
- Se implementarán los procesos en un laboratorio especializado, en donde se realizarán pruebas de funcionamiento de las librerías implementadas en el Sistema Operativo FreeRTOS.

1.4 MARCO TEÓRICO

En esta sección se recopilan los conceptos básicos que serán necesarios para comprender el funcionamiento y las aplicaciones del sistema operativo FreeRTOS, así como la forma correcta de implementación en un microcontrolador para poderlo trabajar de manera similar a un PLC industrial. Comprende también un análisis de los principales bloques industriales que son usados en la programación de los controladores lógicos programables con el fin de utilizarlos en la plataforma de Arduino.

1.4.1 SISTEMA OPERATIVO FREERTOS

El sistema operativo FreeRTOS fue desarrollado en el año 2003 por Richard Barry para posterior a esto ser desarrollada y mantenida por la compañía Real Time Engineers Ltd., la cual estaba a cargo de Richard Barry. El sistema operativo FreeRTOS para los años venideros fue muy exitoso incluso teniendo una alta calificación en las encuestas EETimes

sobre sistemas operativos integrados, por lo que llamó la atención de varias compañías, en el año 2017 la compañía Real Time Engineers Ltd. paso el proyecto del sistema operativo FreeRTOS a Amazon Web Servicies (AWS) [1].

Al igual que FreeRTOS se han desarrollado otros sistemas operativos los cuales son utilizados para diferentes aplicaciones, por lo que se debe evitar confundirlos al momento de utilizarlos, como es el caso del sistema operativo de propósito general (GPOS), el cual presenta algunas diferencias con RTOS, ya que este último está más enfocado a sistemas embebidos y microcontroladores y GPOS se enfoca más a computadoras o teléfonos. Otra principal diferencia es que RTOS es determinístico lo que indica que no se tiene ejecuciones aleatorias a lo contrario de GPOS [2].

FreeRTOS es un sistema operativo en tiempo real el cual permite controlar los tiempos de ejecución de las tareas que se han implementado en un microcontrolador y permite administrar los recursos del hardware para trabajar de manera más eficiente.

Los microcontroladores pueden realizar diferentes tipos de tareas los cuales serán programas que desarrollarán operaciones específicas. Pero cuando se realiza un proceso con un sistema operativo bajo el contexto de multitareas los recursos del hardware del microcontrolador serán compartidos por varios programas por lo que se necesitará administrarlos y darles orden de prioridad, pero se debe tomar en cuenta que dichas tareas únicamente se ejecutaran cuando los recursos del hardware estén disponibles, caso contrario la tarea no será ejecutada [2].

1.4.1.1 Tareas en FreeRTOS [3]

El sistema operativo FreeRTOS es utilizado cuando varias operaciones van a ser ejecutadas al mismo tiempo, pero teniendo en cuenta que los recursos del microcontrolador van a ser compartidos. Las operaciones que se realicen en el microcontrolador se las va a denominar tareas, las cuales van a tener un orden de prioridad donde 0 será la menor prioridad, y su límite máximo será restringido por la cantidad de RAM del microcontrolador por lo que es necesario tener un control adecuado de las mismas.

Los estados de las tareas se dividirán en estado de funcionamiento Running, ready, blocked y suspended. Donde el estado Running implica que se está realizando la tarea, pero los estados no operativos se los puede subdividir tal y como se puede observar en la Figura 1.1.

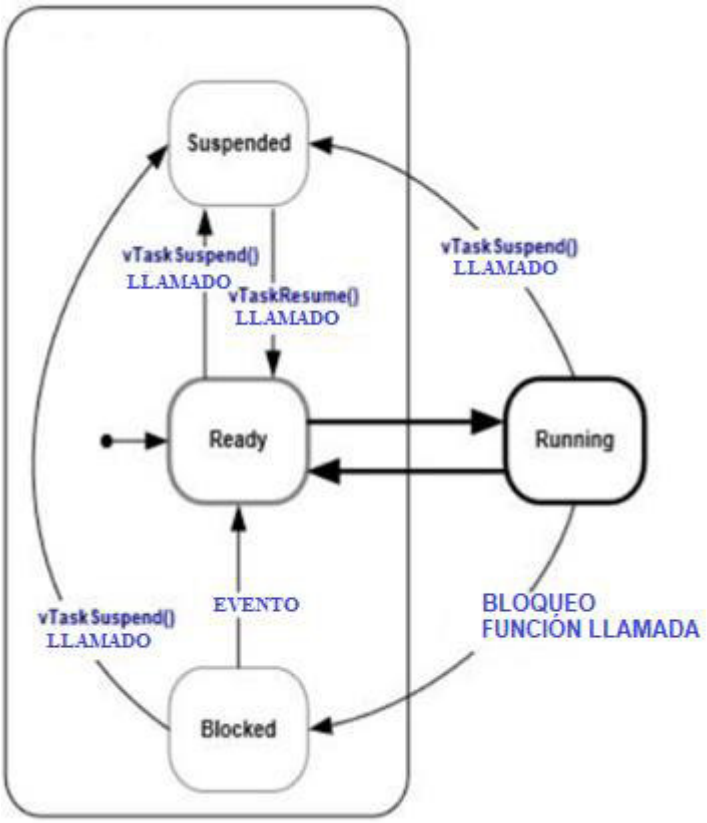


Figura 1.1. Esquema de programación [2]

- Running: Indica que la tarea ya tiene asignado los recursos y se está ejecutando.
- Ready: Indica que una tarea se encuentra lista, pero se encuentra a la espera que "scheduler" le permita entrar a estado Running. Donde scheduler es quien permite pasar una tarea de estado ready a running, pero si varias tareas se encuentran en estado ready, se tomará en cuenta el orden de prioridad para pasar a running.
- Blocked: "scheduler" lleva a las tareas a este estado cuando detecta que se encuentran en un evento temporal como en un delay, interrupción, entre otros, y cuando termina estas acciones "scheduler" los regresa al estado Ready.
- Suspended: Es muy similar al estado blocked, pero la diferencia es que para salir de este estado se lo debe realizar con la función vTaskSuspend() y no con "scheduler" [3].

1.4.1.2 Tareas estáticas en FreeRTOS

En el sistema operativo FreeRTOS, se podrá crear tareas estáticas y dinámicas, y cada una de ella presenta ventajas y desventajas en comparación de la otra, pero lo recomendable es trabajar con tareas estáticas, ya que las dinámicas presentan algunos

problemas de fragmentación de la memoria, debido a esto se recomienda el uso de tareas estáticas, ya que este método ayuda a entregar un resultado correcto y al tiempo indicado.

Pero para trabajar con este método se debe conocer algunos conceptos básicos como la pila, la cual permite guardar variables locales de las tareas, parámetros de llamadas a funciones y dirección de regreso de una función llamada. También se debe conocer el termino bloque de control (TCB), el cual ayuda a guardar la información más importante de cada tarea. Por ejemplo, cuando una tarea se va a dormir debe guardar este estado, para luego devolver dicha información cuando la tarea esta lista para ejecutarse.

Se muestra en la Figura 1.2 un ejemplo de la creación de una tarea estática con la función xTaskCreateStatic, donde se tendrá sus argumentos como ulStackDepth el cual hace las veces de pila.

```
TaskHandle_t xTaskCreateStatic( TaskFunction_t pxTaskCode,
                                const char * const pcName,
                                const uint32_t ulStackDepth,
                                void * const pvParameters,
                                UBaseType_t uxPriority,
                                StackType_t * const puxStackBuffer,
                                StaticTask_t * const pxTaskBuffer );
```

Figura 1.2. Creación de una tarea estática [4]

Pero un aspecto a considerar es el tamaño que se le va a dar a la pila, ya que dependerá del trabajo que deba realizar la tarea, ya que si se escoge una pila con un valor pequeño se corre el riesgo de que se desborde, y si se escoge un valor muy grande se desperdiciaría memoria RAM, por lo que escoger un valor adecuado es de suma importancia. Escoger un valor correcto será resultado de intentar y equivocarse (trial and error) [4].

1.4.1.3 Aplicaciones de FreeRTOS

El sistema operativo FreeRTOS lleva varios años en el mercado por lo que se ha empezado a utilizar en varios sectores debido a las grandes ventajas que otorga, por lo que se las puede encontrar en aplicaciones industriales, productos de consumo, entre otras.

Aplicaciones industriales

A nivel industrial se utiliza dispositivos basados en microcontroladores los cuales permiten generar datos sobre los procesos en los que se estén empleando. Las bombas, sensores, accionadores, utilizan microcontroladores porque pueden desempeñar acciones en tiempo real. Como es el caso de una plataforma petrolífera donde cada una de sus bombas están controladas mediante un microcontrolador. FreeRTOS permite realizar una recopilación de datos sobre el comportamiento del sistema como rendimiento y desgaste, y esto se realiza mediante una conexión directa con la nube, lo que permitirá tomar acciones en tiempo real con plataformas como AWS (amazon web services) para impedir interrupciones [5].

Productos de consumo

FreeRTOS es utilizado por los fabricantes de productos de consumo, como iluminación inteligente, electrodomésticos, entre otros, debido a que ayuda a realizar los mantenimientos de dispositivos basados en microcontroladores y a estandarizar el desarrollo. FreeRTOS es un sistema operativo que es compatible con una gran variedad de hardware de microcontroladores. Por lo que las empresas pueden evitar el desarrollo de software en varias líneas de producto [5].

1.4.2 PLC CRONOS

El PLC cronos fue desarrollado en el año 2019 por Ángel Geovanny Torres Carrión estudiante de la Escuela Politécnica Nacional. Su trabajo se enfocó en presentar el diseño y construcción de un controlador lógico programable (PLC) basado en una plataforma de 32 bits, la interface de programación que utiliza es Arduino IDE.

El dispositivo cuenta con entradas digitales de 0 a 24 voltios, salidas digitales tipo relé, entradas analógicas de 0 a 10 voltios y un puerto RJ45 que permite al usuario comunicarse mediante ethernet. En la Tabla 1.1 se puede observar las características tanto de entradas y salidas con sus respectivos valores [6].

Tabla 1.1. Características del PLC CRONOS [6]

Sección	Característica	Valor
Alimentación	Tensión nominal	24 [VDC]
	Rango de tensión	13 a 28 [VDC]
	Intensidad de entrada típica	200 [mA]
	Intensidad de entrada máxima	600 [mA]
Entradas digitales	Número de entradas	8
	Tensión nominal	24 [VDC]
	Tensión máxima	27 [VDC]
	Señal 1 lógica (min.)	10 [VDC]
	Señal 0 lógica (máx.)	5 [VDC]
	Número de entradas ON simultáneamente	8
Entradas analógicas	Número de entradas	2
	Tipo	Tensión (asimétrica)
	Rango	0 a 10 [V]
	Resolución	8, 9, 10, 11, 12 [bits]
	Tensión máxima	20 [V]
	Tipo de protección	Relé
	Longitud de cable	10 [m], par trenzado apantallado
Salidas digitales	Número de salidas	6
	Tipo	Relé
	Máxima tensión AC	220 [V]
	Máxima tensión DC	24 [V]
	Intensidad máxima	3 [A]
	Resistencia en estado ON	100 [mΩ]
	Retardo de conmutación máx.	10 [ms]
Comunicación	Número de puertos	1
	Tipo	Ethernet
	Transferencia de datos	10/100 Mb/s
	Tipo de cable	CAT5e apantallado
	Protocolos industriales	Modbus TCP IP

1.4.3 BLOQUES DE FUNCIÓN PREDEFINIDOS

Los bloques de función predefinidos son muy utilizados en la programación de PLCs, ya que permiten realizar diversas tareas de una manera sencilla, disminuyendo líneas de código y espacio de memoria. Son bloques que internamente tienen desarrollado una programación la cual les permite realizar tareas específicas como temporizadores, cálculos matemáticos, contadores, entre otros, brindando así mayor facilidad al momento de realizar un código ya que varias acciones están ya programadas y puede ser utilizadas mediante un bloque, los cuales solicitarán algunas acciones de entrada para poder obtener las salidas deseadas.

1.4.3.1 Temporizador de retardo a la conexión (TON)

El bloque de retardo a la conexión permite que la salida se active una vez transcurrido un tiempo establecido por el programador, En la Figura 1.3, se tiene la forma del bloque TON y en la Tabla 1.2 la descripción de pines.

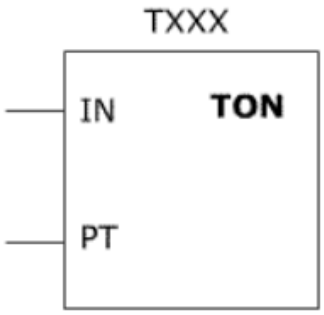


Figura 1.3. Temporizador de retardo a la conexión [7]

Tabla 1.2. Descripción de pines TON [7]

Conexión	Descripción
IN	Cuando IN se active, se iniciará el tiempo de retardo a la conexión
PT	Es el tiempo de retardo a la conexión tras haber sido activado IN
Q	La salida se activará una vez transcurrido el tiempo PT, pero si el parámetro IN sigue activo.

1.4.3.2 Temporizador de retardo a la desconexión (TOF)

El bloque de retardo a la desconexión permite que la salida se mantenga activa durante un periodo de tiempo cuando IN ya se haya desactivado, Se observa en la Figura 1.4, la forma del bloque TOF y en la Tabla 1.3 la descripción de pines.

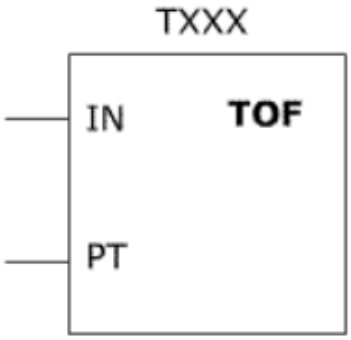


Figura 1.4. Temporizador de retardo a la desconexión [7]

Tabla 1.3. Descripción de pines TOF[7]

Conexión	Descripción
IN	Cuando IN tenga un flanco descendente se iniciará el tiempo para el retardo a la desconexión.
PT	Es el tiempo de retardo a la desconexión tras haber tenido el flanco descendente en IN.
Q	La salida se desconectará una vez transcurrido el tiempo PT.

1.4.3.3 Contador ascendente/descendente (CTUD)

El bloque de función contador ascendente y descendente permite incrementar y disminuir un valor hasta que este sea igual a un valor establecido para poder activar o desactivar su salida, En la Figura 1.3 se muestra la forma del bloque CTUD y en la Tabla 1.4 la descripción de pines.

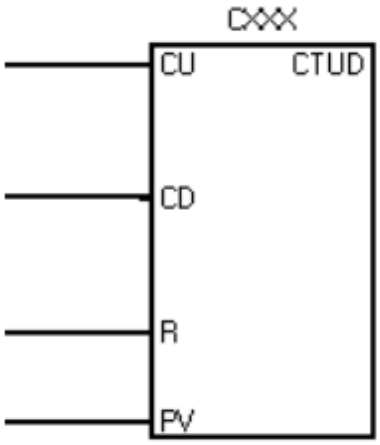


Figura 1.5. Contador ascendente/descendente [8]

Tabla 1.4. Descripción de pines CTUD[8]

Conexión	Descripción
CU	Cuando exista un pulso en CU, el valor del contador aumentara en uno.
CD	Cuando exista un pulso en CD, el valor disminuirá en uno.
R	Cuando exista un pulso en R, reiniciara el contador a cero.
PV	Es un valor establecido por el programador y sirve como referencia.
Q	Cuando el valor del contador sea igual al valor establecido en PV, el bit de activación será 1.

1.4.3.4 Escalar con parámetros (SCP)

Este bloque permite escalar valores que ingresen al PLC y permite obtener salidas escaladas y más comprensibles para los operadores, En la Figura 1.6, se muestra la forma del bloque SCP y en la Tabla 1.5 la descripción de pines.

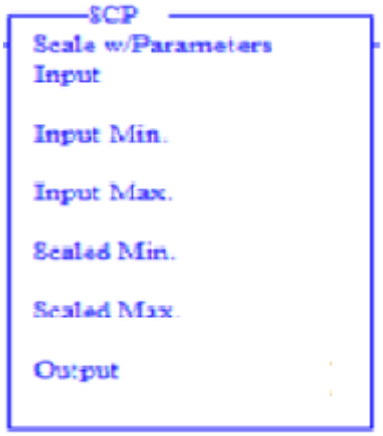


Figura 1.6. Bloque para escalar entradas analógicas [9]

Tabla 1.5. Descripción de pines SCP [9]

Conexión	Descripción
INPUT	Entrada analógico real
INPUT MIN	Valor mínimo real en BCD
INPUT MAX	Valor máximo real en BCD
SCALED MIN	Valor mínimo escalado
SCALED MAX	Valor máximo escalado
OUTPUT	Valor analógico real después de escalar

1.4.4 PROCESOS A SER IMPLEMENTADOS

En este trabajo de integración curricular se trabaja con dos procesos que se ejecutan de manera simultánea con ayuda del sistema operativo FreeRTOS. El primero consiste en el control de temperatura de un horno y el segundo un vaivén continuo del actuador neumático.

Para el primer proceso el PLC utiliza 2 salidas digitales para controlar el ventilador y una niquelina, así como una entrada analógica para medir la temperatura mediante un sensor tipo RTD. El proceso consiste en un control de temperatura on/off con histéresis con su límite superior en 60 grados y su límite inferior en 40 grados. Se inicia con el encendido de la niquelina, la cual una vez encendida empezará a incrementar la temperatura del horno hasta llegar a los 60 grados, en este punto, se desactivará la niquelina y se activará el ventilador para que la temperatura disminuya hasta los 40 grados, en este punto se desactiva el ventilador y volverá a activarse la niquelina, el proceso se repetirá 3 veces.

Para el segundo proceso se utiliza dos salidas digitales y una entrada digital, para controlar una electroválvula, la que estará conectado a un actuador neumático. El proceso consiste

en un vaivén continuo que empieza cuando se oprime el pulsador de marcha/paro, el que activa la electroválvula la que permitirá que el vástago salga del actuador neumático, y utilizando un temporizador se accionará la segunda entrada de la electroválvula lo que permite regresar el vástago a su posición original. El proceso terminará cuando se oprima nuevamente el pulsador de marcha/paro.

1.4.5 SENSORES Y ACTUADORES

Las aplicaciones que controla el PLC utilizan varios sensores y actuadores para que se realice varias tareas como: medir temperatura, disminuir e incrementar la temperatura del horno, entre otras. A continuación, se presentan cada uno de ellos con sus respectivas características.

1.4.5.1 Sensor de temperatura

Es un sensor RTD tipo PT100 que permite medir temperatura y entregar a su salida una señal resistiva, la que será enviada a un acondicionador para cambiar la señal a un voltaje estandar. El sensor se muestra en la Figura 1.7.

Figura 1.7. Sensor RTD [10]

Las características del sensor se presentan en la Tabla 1.6.

Tabla 1.6. Características del ventilador [10]

Características	**Descripción**
Rango de temperatura	-50 a 250 °C
Tipo	PT100
Cables	4 cables conductores
IP	68

1.4.5.2 Ventilador axial

El ventilador axial permite mover caudal de aire en una zona determinada mediante el movimiento de sus paletas, permitiendo que el aire entre y salga siguiendo una trayectoria paralela al eje de la hélice. En la Figura 1.8 se puede observar el ventilador axial.

Figura 1.8. Ventilador [11]

En la Tabla 1.7 se tiene las características del ventilador como el caudal de aire que brinda y las revoluciones por minuto que dará el motor.

Tabla 1.7. Características del ventilador [11]

Características	**Descripción**
Caudal de aire	Mín.: 172 m^3/h (6.074,12 ft^3/h) Máx.: 425 m^3/h (15.008,73 ft^3/h)
Nivel sonoro	Mín.: 105 dB Máx.: 108 dB
Tipo	Axial
Material	Cubos de turbina de acero inoxidable
RPM	2500/2800

1.4.5.3 Niquelina

Son resistencias calefactoras que convierten la energía eléctrica en calor, se utiliza para incrementar la temperatura en un determinado espacio, mediante la obtención de energía calorífica a través de las resistencias. La niquelina se muestra en la Figura 1.9.

Figura 1.9. Niquelina [12]

En la Tabla 1.8 se muestra las características de la niquelina que será utilizada para la aplicación.

Tabla 1.8. Características de la niquelina [12]

Características	**Descripción**
Tensión	220V
Potencia	300w
Rango de temperatura	150

1.4.5.4 Cilindro Neumático

Es un actuador lineal que puede ser de simple o de doble efecto, utilizado para generar trabajo a partir de energía neumática. En la Figura 1.10 se muestra un cilindro neumático.

Figura 1.10. Cilindro neumático [13]

En la Tabla 1.9 se muestra las características del cilindro neumático que será utilizado para la aplicación.

Tabla 1.9. Características de cilindro neumático [13]

Características	**Descripción**
Consumo de aire	0,105
Diámetro vástago	12 mm
Presión de trabajo	10 bar max
Temperatura de trabajo	-20 a 80 C

1.4.5.4 Electroválvula

Es una válvula que permite al paso de aire hacia una de sus salidas en función de una señal eléctrica de pilotaje. Se utilizan para controlar a cilindros neumáticos. En la Figura 1.11 se muestra la electroválvula que se usará en la aplicación.

Figura 1.11. Electroválvula [14]

En la Tabla 1.10 se muestra las características de la electroválvula que será utilizada para la aplicación.

Tabla 1.10. Características de la electroválvula [14]

Características	**Descripción**
Tensión	24 V
caudal	200 a 700
Material	Aluminio

1.4.5.5 Pulsador

Un pulsador es un dispositivo mecánico que acciona un contacto NA o NC al ser presionado. Se utiliza para enviar una señal a una entrada digital, que puede estar en un microcontrolador o controlador lógico programable, lo que permite ejecutar alguna acción.

Figura 1.12. Pulsador [15]

En la Tabla 1.11 se muestra las características del pulsador que será utilizada para la aplicación.

Tabla 1.11. Características del pulsador [15]

Características	**Descripción**
Tensión máxima	28 V
Corriente máxima	3 A
Temperatura de operación	: -40 °C to +85 °C

1.4.5.6 Acondicionador

Un acondicionador es un dispositivo que permite ingresar valores no estandarizados de un sensor y entregar a su salida un valor estándar de voltaje o corriente. La Figura 1.13 muestra un acondicionador de 3 hilos con valores de ingreso resistivos y salida de voltaje.

Figura 1.13. Acondicionador[16]

En la Tabla 1.12 se muestra las características del acondicionador que será utilizada para la aplicación.

Tabla 1.12. Características del pulsador [16]

Características	**Descripción**
RTD	100 ohm
Salida	0-10V
hilos	2 o 3

2 METODOLOGÍA

En el presente trabajo de titulación se emplean las técnicas de consulta y experimentación, para recopilar la información necesaria que ayude al desarrollo de librerías e implementación del sistema operativo.

El trabajo de titulación consta en tres fases. La primera fase es la teórica, la cual se desarrolló en el primer capítulo, donde se detalla el sistema operativo FreeRTOS, y las formas en las que se debe crear y trabajar las tareas, además se detalló la información sobre los bloques de función más utilizados en la programación industrial los cuales serán útiles para la creación de librerías y desarrollo de una aplicación industrial. La segunda fase presentada en el capítulo 2 es el diseño, donde se describirá las funciones que se utilizará para crear tareas estáticas y el desarrollo de las librerías las cuales serán utilizadas en el sistema operativo para desarrollar una aplicación industrial. La fase de implementación, también descrita en el segundo capítulo, describe la concatenación de los algoritmos y el PLC cronos en un proceso industrial desarrollado en un laboratorio especializado. Por último, en el capítulo 3 se presentará la fase de resultados donde se mostrará las pruebas de funcionamiento del sistema y su efectividad en la industria.

2.1 DESARROLLO DE HARDWARE

En esta sección se describe los circuitos y las características del PLC que se utilizará, los dispositivos que serán utilizados como entradas y salidas, y la manera en que se los conectara al PLC. La Figura 2.1, muestra la arquitectura del hardware, donde se observa cada uno de los elementos que forman parte del sistema que constituye la aplicación que se desarrolla en el presente trabajo de integración curricular con la finalidad de probar el sistema operativo FreeRTOS en el PLC Cronos controlando una aplicación industrial trabajando con dos procesos simultáneos.

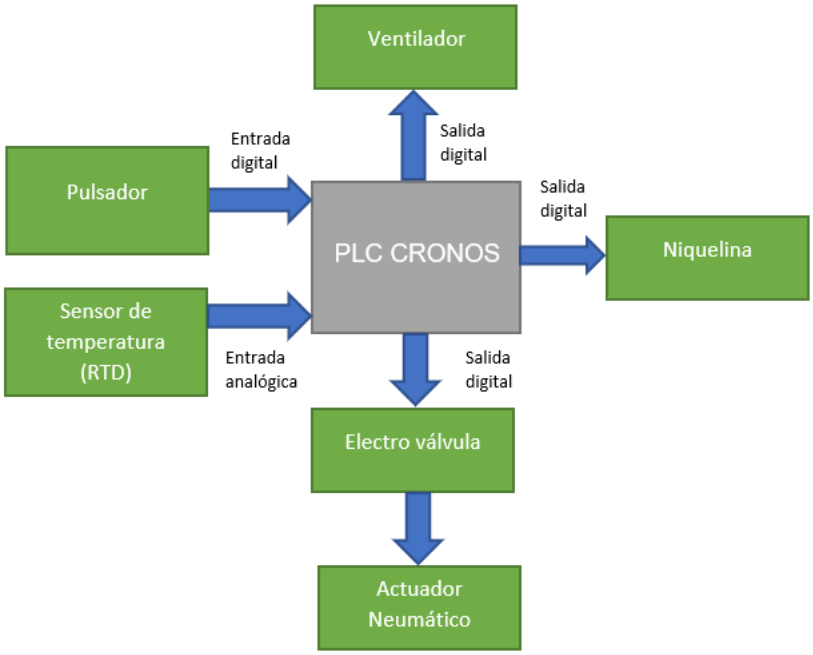


Figura 2.1. Arquitectura del hardware

2.1.1 PLC CRONOS

El controlador lógico programable cronos, fue desarrollado por la Escuela Politécnica Nacional, con la finalidad de poder desarrollar procesos industriales, partiendo desde una programación de Arduino, la cual facilitaría su manejo. Fue desarrollado a partir de una plataforma STM32F103C [2].

2.1.1.1 Plataforma STM32F103C

El microcontrolador STM32F103C es una tarjeta de desarrollo de alto rendimiento fabricado por la familia STM, la que brinda varias características en un núcleo tipo RISC, que trabaja con una frecuencia de 72 MHz a 32 bits. En la Figura 2.2 se muestra el diagrama de pines del microcontrolador el cual fue la base para desarrollar el PLC.

Pero su característica más importante para este trabajo es que dicha plataforma puede trabajar con el software de desarrollo Arduino IDE, lo que permitirá utilizar las librerías desarrolladas para trabajar con el Controlador lógico programable.

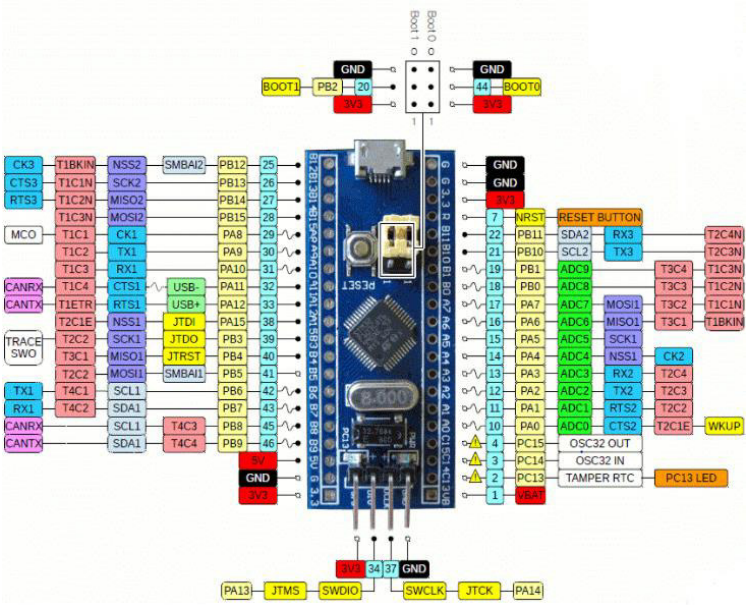


Figura 2.2. Diagrama de pines [17]

2.1.1.2 Entradas digitales

El controlador lógico programable Cronos, cuenta con 8 entradas digitales, las cuales se diseñaron para trabajar a 24 V de voltaje continuo, mientras que al microcontrolador le llegarán 3.3 V. En la Figura 2.3 se muestra el circuito utilizado por el PLC para las entradas digitales, Se utiliza un optoacoplador para separar los voltajes de entrada y el voltaje de ingreso al Pin del microcontrolador [2].

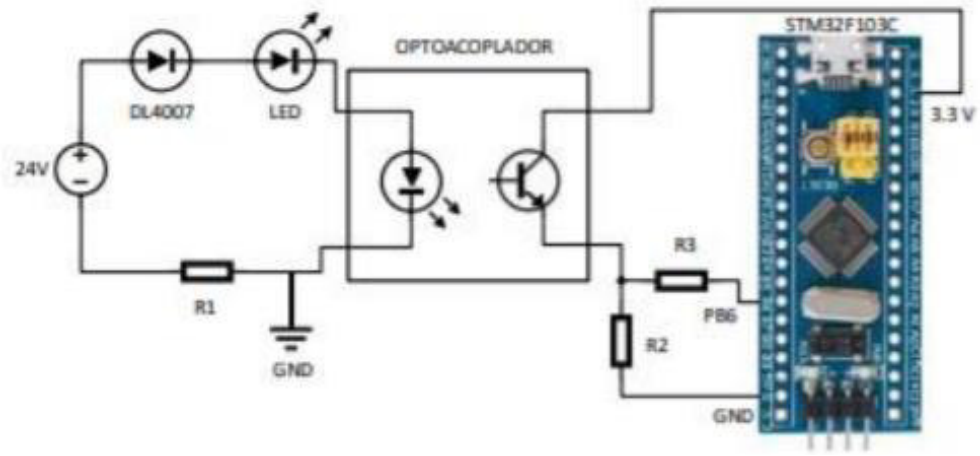


Figura 2.3. Circuito de entrada digital [6]

2.1.1.3 Entradas analógicas

El controlador lógico programable Cronos, tiene dos entradas analógicas, las cuales trabajan con el estándar de 0 a 10 V. En la Figura 2.4, se muestra el circuito utilizado para las entradas analógicas [2].

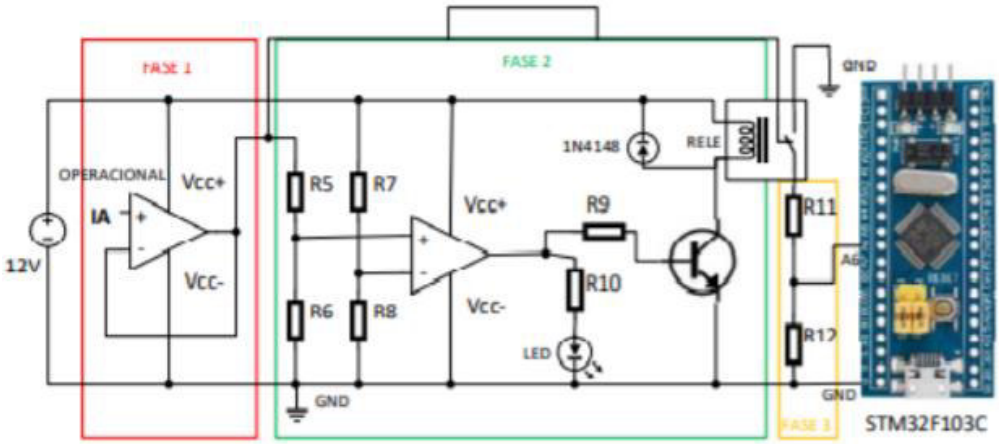


Figura 2.4. Circuito de entradas analógicas [6]

2.1.1.4 Salidas digitales

El controlador lógico programable Cronos, cuenta con 6 salidas digitales, las cuales utilizan relés para brindar un contacto seco, tal y como se muestra en la Figura 2.5 [2].

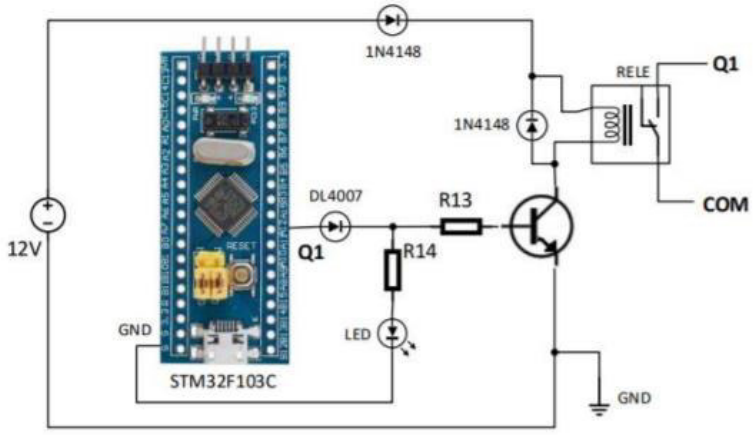


Figura 2.5. Circuito de salida digital [6]

2.1.2 CONEXIÓN SENSORES Y ACTUADORES

En la Figura 2.6 se puede observar la conexión del PLC con los sensores y actuadores, donde el sensor como ya se mencionó tienen una salida resistiva y mediante un acondicionador se obtiene una salida estándar de 0 a 10 V para que pueda ingresar al PLC. Para el caso de los actuadores neumáticos se tendrá una salida de voltaje desde el PLC y llegará a un relé el cual dará una salida de 24 V lo que alimentará a la electroválvula y permitirá accionar el actuador neumático. Para el caso de la niquelina y el ventilador se tendrá de manera similar, pero se usará un relé que permite tener una salida de 110 V, lo cual permitirá alimentar la niquelina y el ventilador.

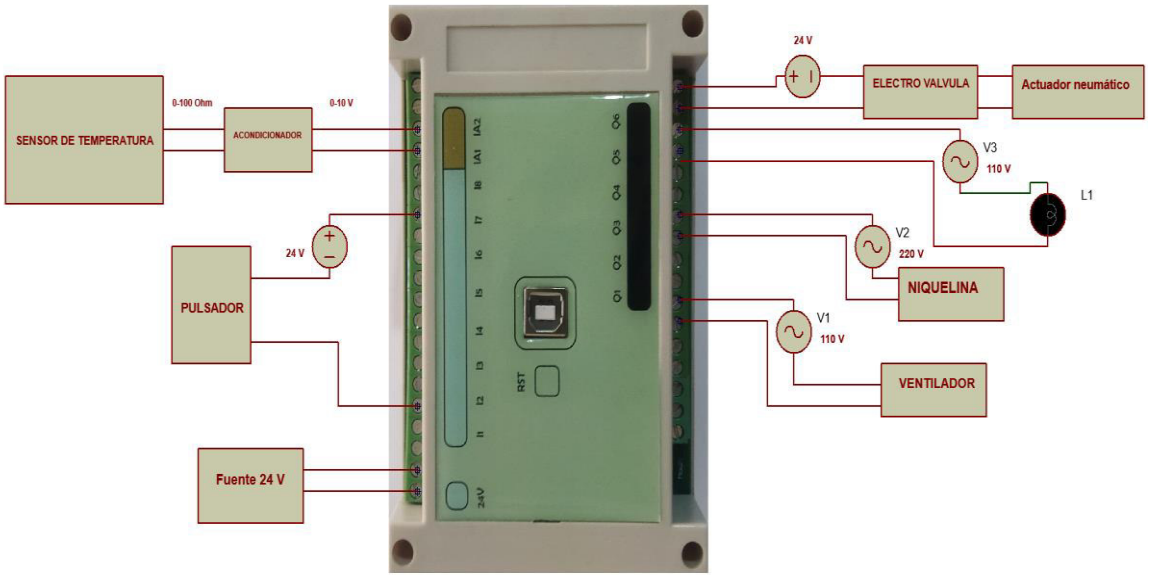


Figura 2.6. Conexión PLC

2.2 DESARROLLO DEL SOFTWARE

En esta sección se describe los conceptos teóricos utilizados para el desarrollo de los algoritmos que serán utilizados en el PLC Cronos, desarrollado por la Escuela Politécnica Nacional.

Como se puede observar en la Figura 2.7 se tendrá varios algoritmos los cuales serán implementados en el PLC Cronos, en los que se encuentra las librerías de los bloques de función, el sistema freeRTOS y la creación de tareas en el sistema, los cuales ayudarán a desarrollar aplicaciones industriales como control de temperatura, control de actuadores, entre otros.

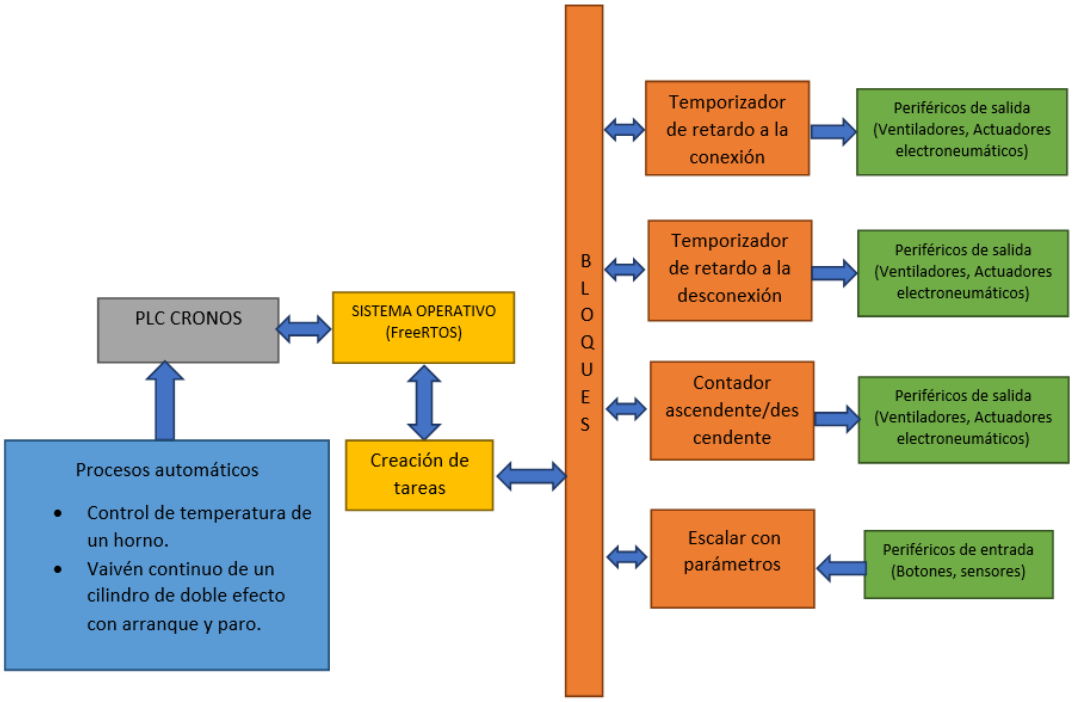


Figura 2.7. Arquitectura del software

2.2.1 SISTEMA OPERATIVO FreeRTOS

Como ya se mencionó el sistema operativo FreeRTOS brinda ventajas al momento de realizar un proceso donde se tenga varias tareas y se requiera de mayor exactitud en cuanto a tiempo y orden de prioridades.

Una vez que se tiene claro las tareas que se va a realizar, el siguiente paso sería registrarlas en el sistema operativo, lo cual se denominará crear tareas, y es en este punto donde se debe tomar en cuenta la forma en la que se creará y los parámetros para su creación.

2.2.1.1 Configuración de FreeRTOS

El sistema operativo FreeRTOS se configura con mucha facilidad en el archivo FreeRTOSConfig.h, este se lo encuentra en la librería que se instala para poder utilizar el sistema operativo en Arduino. Y es recomendable realizar las configuraciones necesarias para poder realizar un proceso de una manera adecuada.

En la Tabla 2.1 se presenta algunos de los parámetros más importantes al momento de configurar el sistema operativo, únicamente habilitando funciones que se utilizarán y así evitar el uso innecesario de memoria, por lo que se debe tener claro el proceso a realizar antes de empezar a configurar el sistema operativo.

Tabla 2.1. Parámetros de configuración [5]

Parámetro	Descripción
configSUPPORT_STATIC_ALLOCATION	Esta constante debe estar en uno para habilitar las tareas estáticas.
configMAX_PRIORITIES.	En esta constante se establece el número máximo de prioridades que se tendrá en el proceso.
configMAX_TASK_NAME_LEN	Aquí se establece el número máximo de caracteres para el nombre de las tareas.
INCLUDE_vTaskDelay	Esta constante debe estar en uno para habilitar el uso de la función TaskDelay en el programa.

2.2.1.1 Desarrollo del algoritmo de creación de tareas

Una vez se tenga claro el número de tareas que tiene el proceso, se las debe registrar en el sistema operativo, para lo cual se utiliza la función xTaskCreateStatic**(),** que permite crear tareas de manera estática, lo que implica que se podrá variar la pila y el TCB, como se presenta en la Figura 2.8, se tendrán varios argumentos que servirán para configurar las diferentes tareas del proceso.

```
TaskHandle_t xTaskCreateStatic( TaskFunction_t pxTaskCode,
                                const char * const pcName,
                                const uint32_t ulStackDepth,
                                void * const pvParameters,
                                UBaseType_t uxPriority,
                                StackType_t * const puxStackBuffer,
                                StaticTask_t * const pxTaskBuffer );
```

Figura 2.8. Argumentos de xTaskCreateStatic [18]

En la Tabla 2.2 se tiene varios parámetros los cuales se deben llenar con criterio para evitar errores o desperdicio de recursos, por lo que se debe conocer ciertos términos como la pila, la que permite: guardar variables locales de las tareas, parámetros de llamadas a funciones y dirección de regreso de una función llamada. Pero se debe tener cuidado con el tamaño que se le asigne a la pila, ya que depende del trabajo que este realizando la tarea, porque, escoger un tamaño ideal es de suma importancia. Escoger un tamaño superior se estaría desperdiciando memoria RAM, mientras que un tamaño menor se corre el riesgo que se desborde la pila. Por lo que el valor correcto va a requerir realizar pruebas donde se ponga números y se verifique el número que sea adecuado, a este método se le denomina intentar y equivocarse (trial and error).

También se debe conocer el término bloque de control (TCB), el que ayuda a guardar la información más importante de cada tarea. Por ejemplo, cuando una tarea se va a dormir debe guardar este estado, para luego devolver dicha información cuando la tarea esta lista para ejecutarse.

Tabla 2.2. Descripción de los argumentos [18]

Argumentos	Descripción
pxTaskCode	Es el nombre de la función que se implementa a la tarea y se le agrega al nombre la partícula _task para poder diferenciarla de las demás funciones.
pcName	Es el nombre que se le dará a la tarea.
ulStackDepth	Número de elementos en la pila.
pvParameters	Se utiliza si se requiere pasarle valores a la tarea que podrían ser desde un simple entera hasta un valor compuesto, pero en caso de no necesitarlo se deberá escribir el valor NULL.
uxPriority	Se utiliza para dar orden de prioridad a las tareas, siendo cero el nivel con menos prioridad.
puxStackBuffe	Es la pila
pxTaskBuffe	Es la dirección del TCB

El segundo paso es utilizar la función xCreateTask, que permite colocar tareas en una cola de prioridad. En la Figura 2.9 se presenta los argumentos que contiene dicha función.

```
xTaskCreate(MyTask_pointer, "task_name", 100, Parameter, Priority, TaskHandle);
```

Figura 2.9 Función xCreateTask [19]

MyTask_pointer: Este argumento es un puntero que ayudará a identificar la tarea y llamarla.

task_name: Este argumento es el nombre que se le asignará a la tarea.

StackDepth: Este argumento representa el tamaño de la pila, la cual, como ya se mencionó se la debe escoger dependiendo del tamaño y complejidad de la tarea, para este ejemplo se colocó 100.

Parámetro: Este argumento es un puntero a una variable que la tarea puede recibir.

Priority: Este argumento permite establecer la prioridad de las tareas, siendo cero la prioridad más baja.

TaskHandle: Este argumento permite de ser necesario cambiar las características de la función, pero en caso de no ser utilizado se coloca el valor NULL.

Una vez definido el número de tareas y sus características, se debe colocar los algoritmos de cada tarea, para esto se utiliza la función Static void, la que permite almacenar el código de cada tarea, como se puede observar en la Figura 2.10, se tendrá el puntero de la tarea (MyTask1), la que se nombrará en la función de la Figura 2.9 en el argumento MyTask_pointer, así se podrá llamar a cada una de las tareas dependiendo de su orden de prioridad.

```
static void MyTask1(void* pvParameters)
{

  while(1)

  {
```

Figura 2.10 Función Static void [19]

2.2.2 BLOQUES DE FUNCIÓN PREDEFINIDOS

En el presente apartado se detallará 4 bloques de función predefinidos, de los cuales dos son temporizadores, el tercero un contador y el cuarto permitirá escalar variables del proceso.

2.2.2.1 Temporizador de retardo a la conexión (TON)

Al entrar en funcionamiento la librería, utiliza el parámetro para el tiempo de retardo con el que se desea trabajar. Al recibir una señal digital el PLC podrá iniciar el tiempo de retardo y una vez terminado el tiempo establecido, el PLC energizará un pin de salida para poder activar una parte del proceso. En la Figura 2.11 se muestra el diagrama de flujo del temporizador.

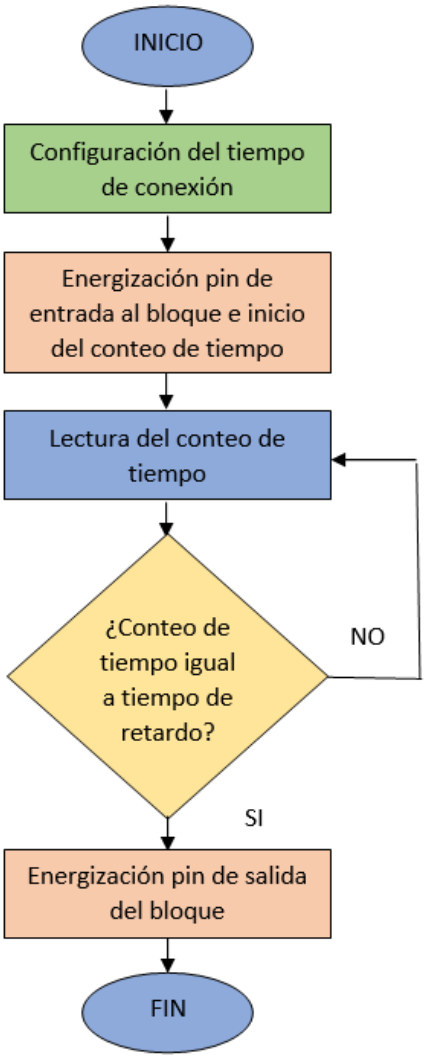


Figura 2.11. Diagrama de flujo del temporizador TON

2.2.2.2 Temporizador de retardo a la desconexión (TOF)

Al entrar en funcionamiento la librería, utiliza el tiempo de retardo con el que se desea trabajar, pero a diferencia con el temporizador TON, el pin de salida del bloque ya está energizado y su función es desactivarlo una vez transcurrido este tiempo. En la Figura 2.12 se muestra el diagrama de flujo del temporizador TOF.

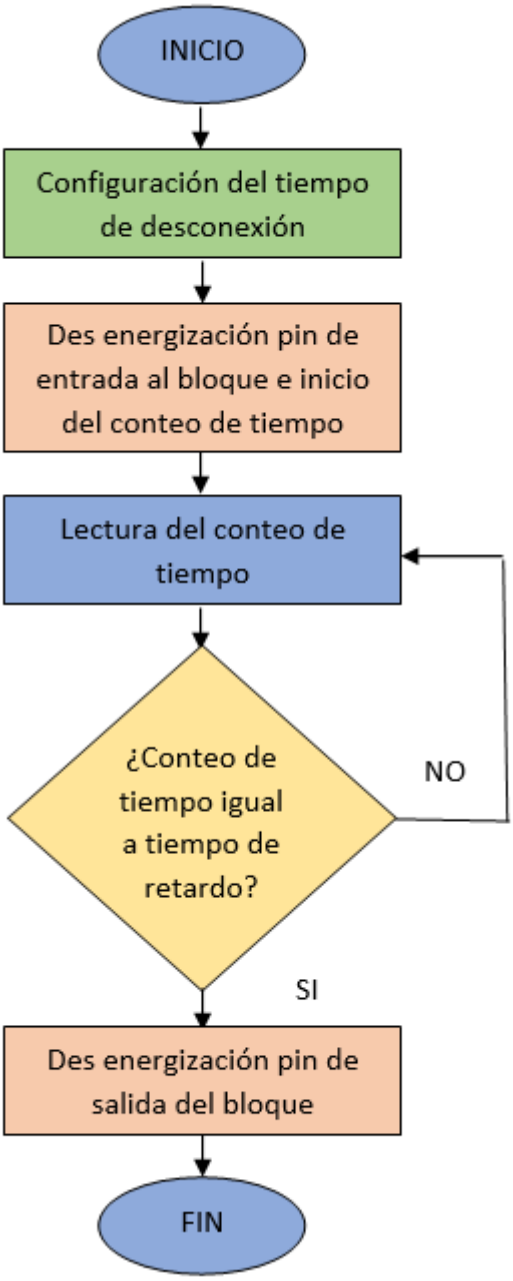


Figura 2.12. Diagrama de flujo del temporizador TOF

2.2.2.3 Contador ascendente/descendente (CTUD)

Al empezar a funcionar la librería, se dará pulsos a la entrada del bloque, el cual incrementará o disminuirá un contador, y cuando este llegue al valor máximo configurado, se energizará la salida del bloque. En la Figura 2.13 se muestra el diagrama de flujo del bloque CTUD.

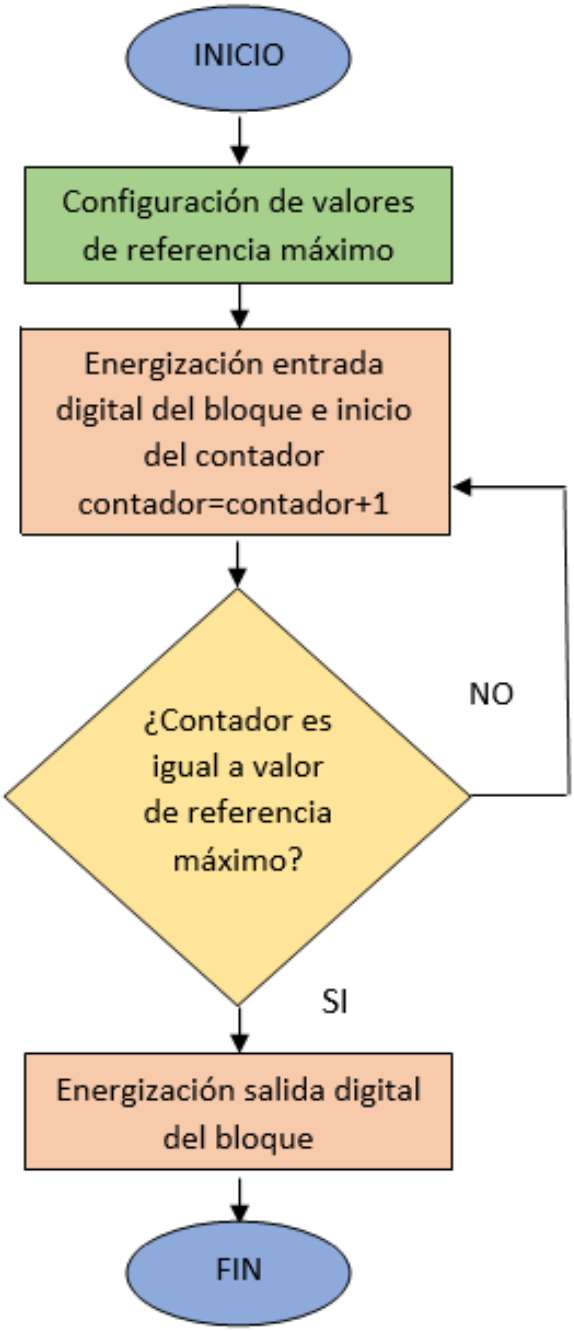


Figura 2.13. Diagrama de flujo del contador ascendente/descendente

2.2.2.4 Escalar con parámetros (SCP)

El controlador lógico programable recibirá variables analógicas, las cuales se requiere escalar para poder ser entendidas por un operador o una persona externa al proceso. Por lo que el bloque permitirá que ingresen los valores máximos y mínimos obtenidos por los

sensores y escalarlos. Obteniendo a la salida valores máximos y mínimos escalados. En la Figura 2.14 se presenta el diagrama de flujo del Escalador SCP.

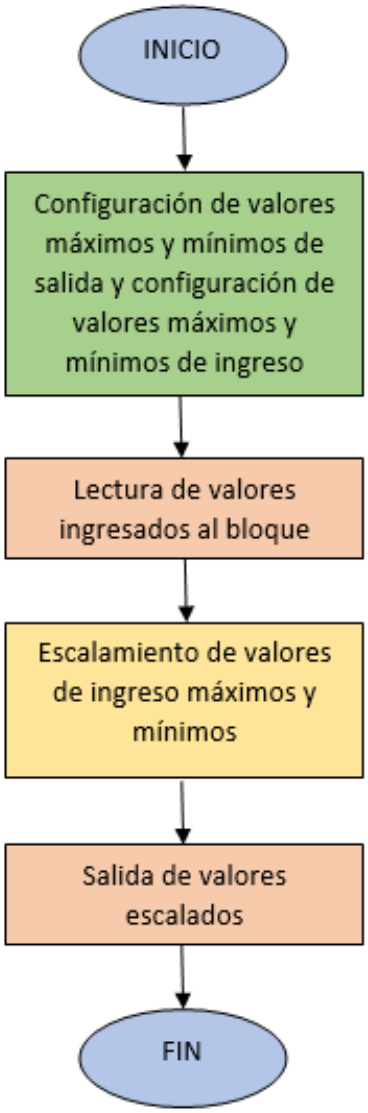


Figura 2.14. Diagrama de flujo de SCP

En el capítulo 3 se presenta las pruebas que se realizó a las librerías y al sistema operativo FreeRTOS, también se muestra los resultados obtenidos al implementarlos en los procesos automáticos, y las conclusiones que se llegó con este trabajo de titulación.

3 RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

En este capítulo se presenta las pruebas realizadas al prototipo del presente trabajo de integración curricular. Primero se muestra las pruebas del sistema operativo FreeRTOS en el controlador lógico programable CRONOS, donde se verifica su funcionamiento y la forma en la que trabaja con diferentes procesos, En segundo lugar, se verifica el funcionamiento de los algoritmos de los bloques de función predefinidos detallando su aplicación en cada proceso que se haya realizado en el PLC.

Luego de analizar el desempeño individual de cada algoritmo, se procede a realizar pruebas de funcionamiento global donde ya se incluirá el sistema operativo y las librerías diseñadas, los cuales serán utilizadas en dos procesos y se evaluará su desempeño en un laboratorio especializado.

3.1 PRUEBAS

En esta sección se presenta cada una de las pruebas realizadas para verificar el funcionamiento de los algoritmos utilizados en el presente trabajo de titulación.

3.1.1 PRUEBA DE LA LIBRERÍA DEL TEMPORIZADOR DE RETARDO A LA CONEXIÓN (TON)

Para iniciar con la prueba del temporizador de retardo a la conexión, se utiliza una salida y una entrada digitales del PLC, y se realiza la configuración del tiempo de retardo, la que está en 3 segundos para la prueba. En la Figura 3.1 se muestra el monitor serial donde se encuentra los resultados de la prueba, y como se muestra en la Figura los primeros ciclos no existe ningún cambio en la entrada y en la salida, pero cuando la entrada se energiza cambia su estado a uno y empieza el conteo del tiempo de retardo y pasado tres ciclos se energizará la salida, comprobando que se respeta el tiempo de retardo a la conexión antes de energizar.

Figura 3.1 Captura del monitor serias durante la prueba del temporizador TON

La Figura 3.2 muestra la conexión que tiene el PLC cronos para la prueba de retardo a la conexión y como se envía un voltaje a la entrada digital para empezar el conteo del tiempo de retardo, y una vez transcurrido dicho tiempo como se energiza la salida digital del PLC.

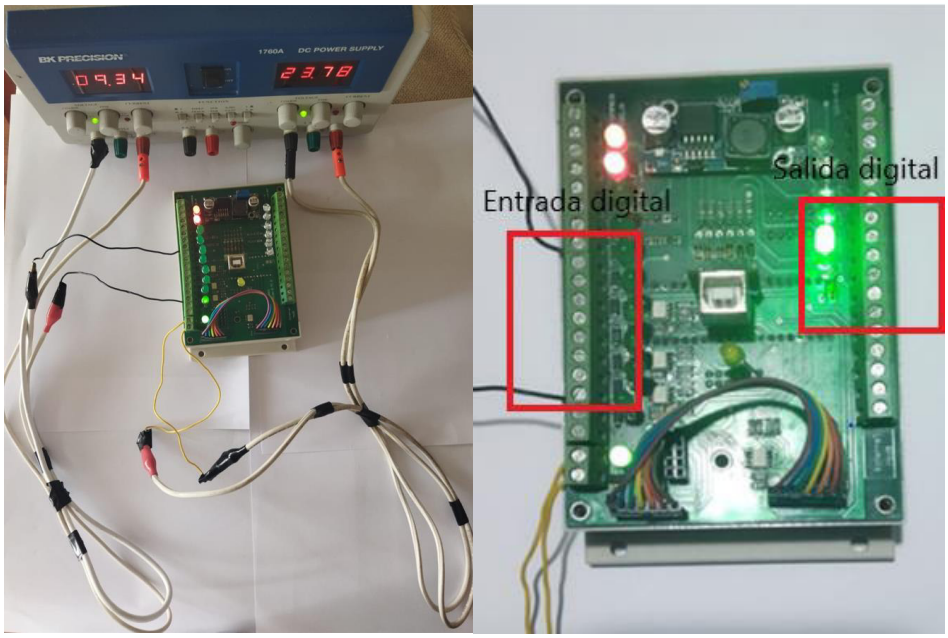


Figura 3.2 Conexión PLC

3.1.2 PRUEBA DE LA LIBRERÍA DEL TEMPORIZADOR DE RETARDO A LA DESCONEXIÓN (TOFF)

Para iniciar con la prueba del temporizador de retardo a la desconexión, se utiliza una salida digital y una entrada digital del PLC, y se realiza la configuración del tiempo de

retardo, la que está en 3 segundos para la prueba. En la Figura 3.3 se muestra el monitor serial donde están los resultados de la prueba, y como se muestra, se tiene en los primeros ciclos la entrada y la salida energizada, y una vez la entrada sea des energizada trascurrirá 3 segundo donde la salida estará en cero, y una vez trascurridos estos ciclos la salida también se des energiza.

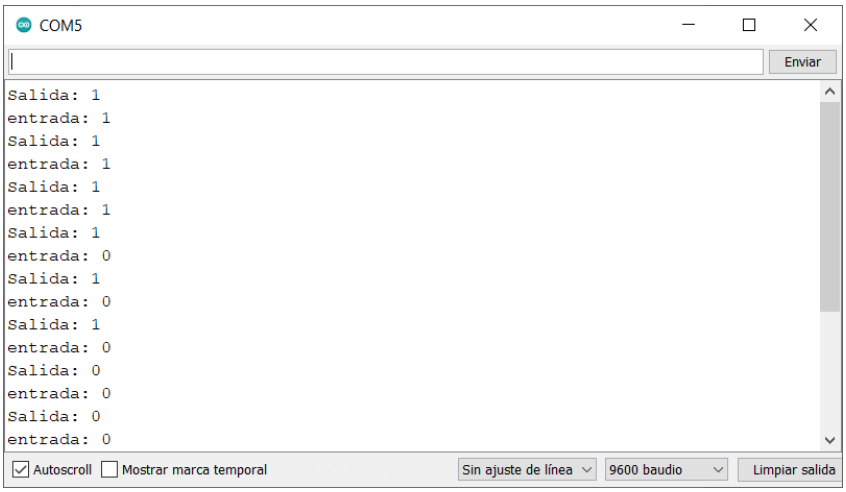


Figura 3.3 Captura del monitor serias durante la prueba del temporizador TOF

Para este caso se tendrá la misma conexión de la Figura 3.2, donde se tenía la entrada y la salida digital.

3.1.3 PRUEBA DE LA LIBRERÍA DEL CONTADOR ASCENDENTE/DESCENDENTE (CTUD)

Para la prueba del contador ascendente/descendente, se utiliza tres pulsadores y una luz piloto para poder mostrar cuando la salida se activa. Para esta prueba se configuró el contador en dos y se utilizó el botón verde de la izquierda para aumentar el contador, el botón rojo del medio para disminuir el contador y el botón verde de la derecha para resetear el contador. Se muestra en la Figura 3.4 la conexión para la prueba y el proceso de encendido de la luz piloto una vez el contador haya llegado al valor establecido.

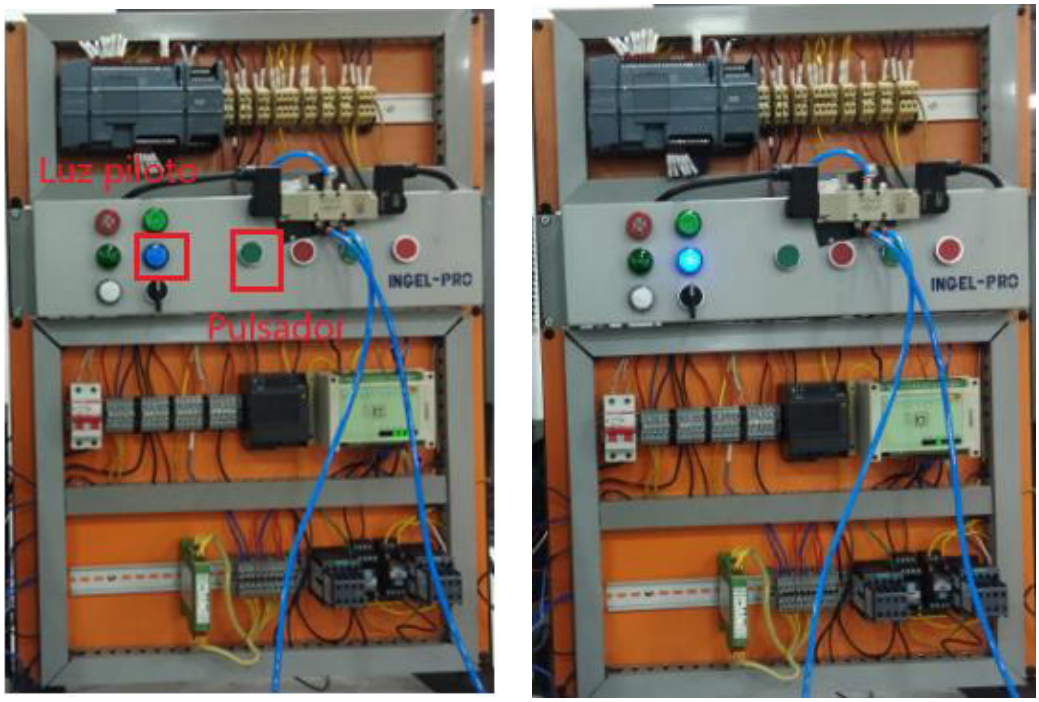


Figura 3.4 Tablero donde se realizó la prueba del contador ascendente/descendente

En la Figura 3.5 se muestra la prueba realizada cuando se aumenta el contador con el pulsador y este llega al número establecido, el cual se configuró en 2. Permitiendo que la salida de energice una vez alcanzado dicho valor y se encienda la luz piloto mostrada en la Figura 3.4.

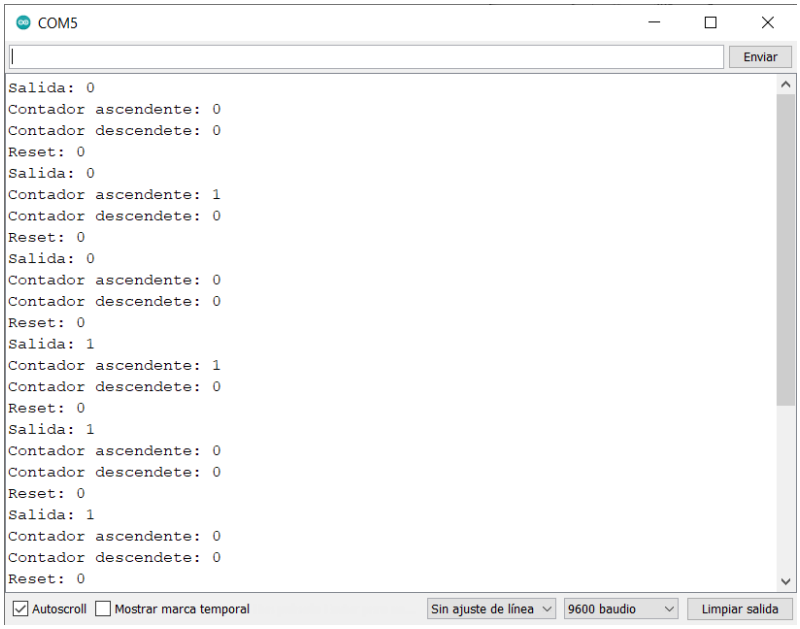


Figura 3.5 Captura del monitor serias durante la prueba del contador ascendente/descendente

3.1.4 PRUEBA DE LA LIBRERÍA ESCALAR CON PARÁMETROS (SCP)

Para realiza la prueba de la librería SCP, se utilizó una entrada analógica del PLC, la que está conectada a una fuente que simula el ingreso de los 10 voltios que dará el sensor de temperatura, y como se muestra en la Figura 3.6 se tiene la conexión con un voltaje inicial de 2.28 voltios, los cuales ingresan al PLC y se realiza la transformación a grados centígrados, los que se muestran en el monitor serial de Arduino.

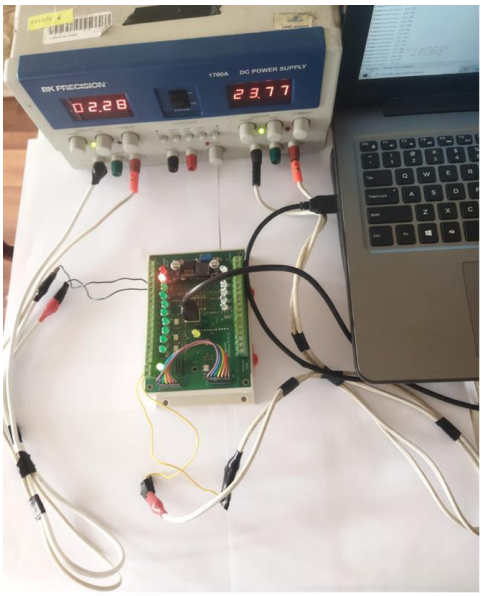


Figura 3.6 Conexión de la prueba

En la Figura 3.7 se muestra los valores de temperatura en grados centígrados, los cuales son resultado del escalamiento que se realizó al ingresar el voltaje de 0 a 10 V en la entrada analógica. En este caso se muestra los valores de temperatura a un voltaje de 2.28 V.

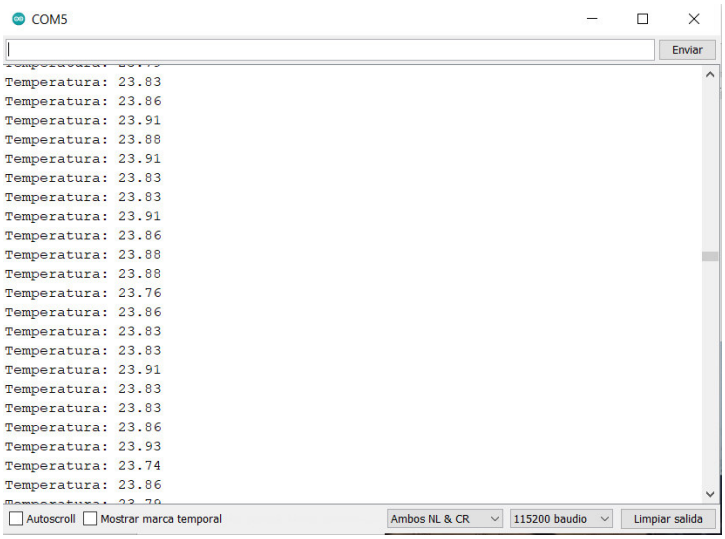


Figura 3.7 Valores obtenidos de temperatura

Para la segunda prueba se realizó un cambio de voltaje al ingreso de la entrada analógica del PLC, ahora con un voltaje de 4.64 V con el fin de verificar los nuevos valores de temperatura y asi comprobar que la conversión de voltaje a temperatura esté funcionando correctamente. En la Figura 3.8 se muestra el ingreso de 4.64 voltios a la entrada analógica del PLC.

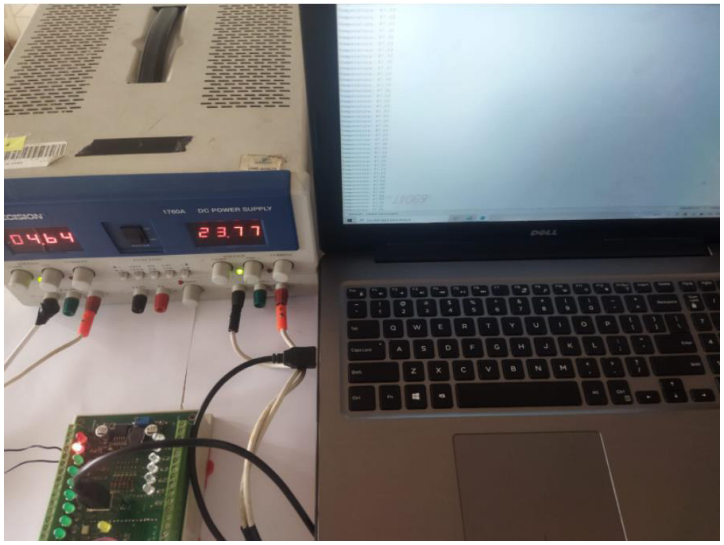


Figura 3.8 Prueba de la librería a un voltaje de 4.64 V

En la Figura 3.9 se muestra los valores de temperatura que se tendrán al enviar a la entrada analógica del PLC un voltaje de 4.64 voltios.

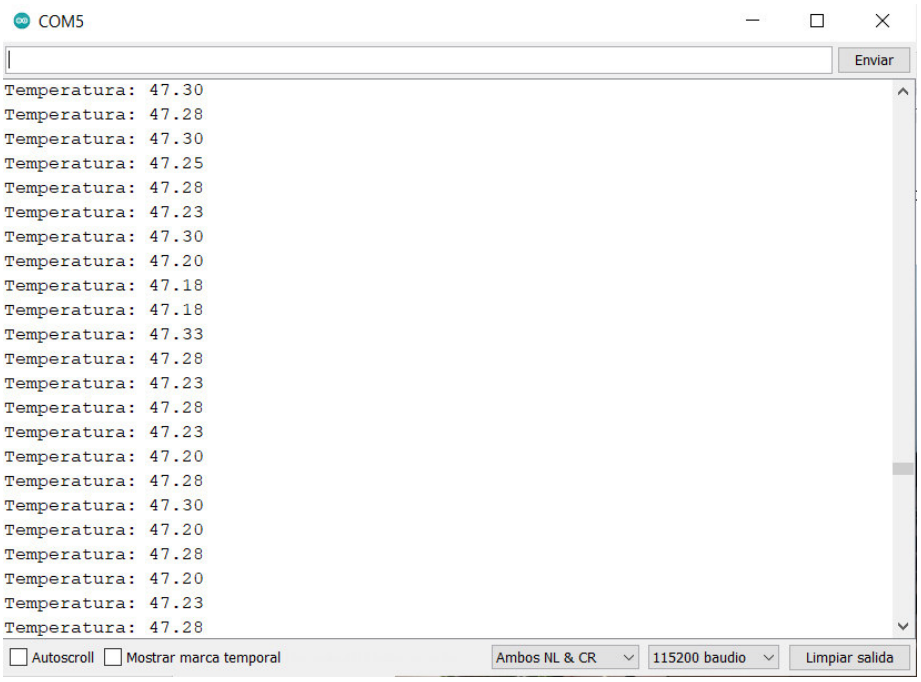


Figura 3.9 Valor de la temperatura a un voltaje de 4.64 V

Para verificar de mejor forma la conversión que se está realizando, en la Tabla 3.1 se tiene varios valores tomados de temperatura con su respectivo voltaje de ingreso.

Tabla 3.1 Valores de voltaje y temperatura

Voltaje	Temperatura
3.03	31.75
5.11	52.40
6.75	68.84
8.56	86.52
10	99.96

3.1.5 PRUEBA DEL SISTEMA OPERATIVO FREERTOS

Para iniciar con las pruebas individuales del sistema operativo FreeRTOS, es necesario verificar su funcionamiento en el microcontrolador (Stm32F103c), el cual es usado por el controlador lógico programable Cronos.

Para la prueba inicial se utiliza 4 periféricos que trabajan de manera secuencial, teniendo en el primer periférico una pantalla lcd 16x2 que muestra la distancia medida por un sensor ultrasónico, el segundo periférico tiene 8 leds controlados por un potenciómetro, el tercer periférico tiene 3 leds que encienden y apagan con diferentes rangos de tiempo y un último periférico tiene un servomotor el cual ira cambiando su ángulo de giro a medida que se mueva el potenciómetro.

Se realiza la prueba variando los parámetros como la distancia al sensor ultrasónico, y posición de los potenciómetros. Se muestra en la Figura 3.10 los valores que toma la distancia, el número de leds que se encienden al variar el potenciómetro, el ángulo de giro el motor y el led que se encuentra encendiendo y apagando.

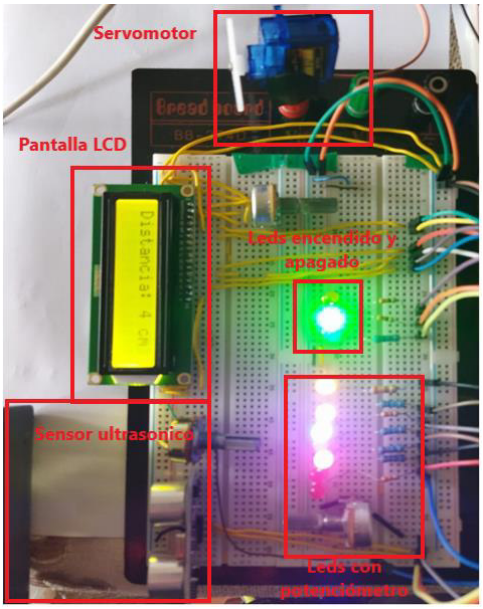


Figura 3.10 Prueba de periféricos sin FreeRTOS

Como segunda parte, se vuelve a variar los parámetros mencionados y se toma de nuevo los valores. Se muestra en la Figura 3.11 los resultados obtenidos, donde se muestra que únicamente los leds de encendido y apagado cambiaron sus valores, apagándose el led que estaba encendido, mientras que los otros periféricos tardaron un poco más de tiempo para cambiar sus valores.

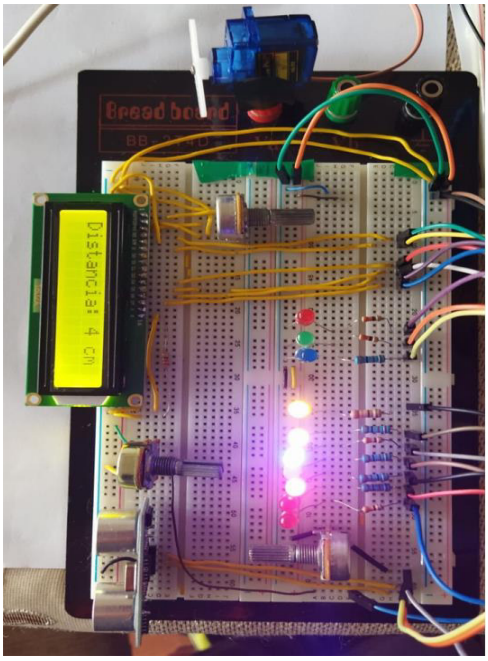


Figura 3.11 Prueba de periféricos sin FreeRTOS

Se muestra en la Figura 3.12, el momento que cambiaron los valores de los otros periféricos, como el número de leds encendidos, la distancia mostrada por la pantalla LCD y la posición del servomotor.

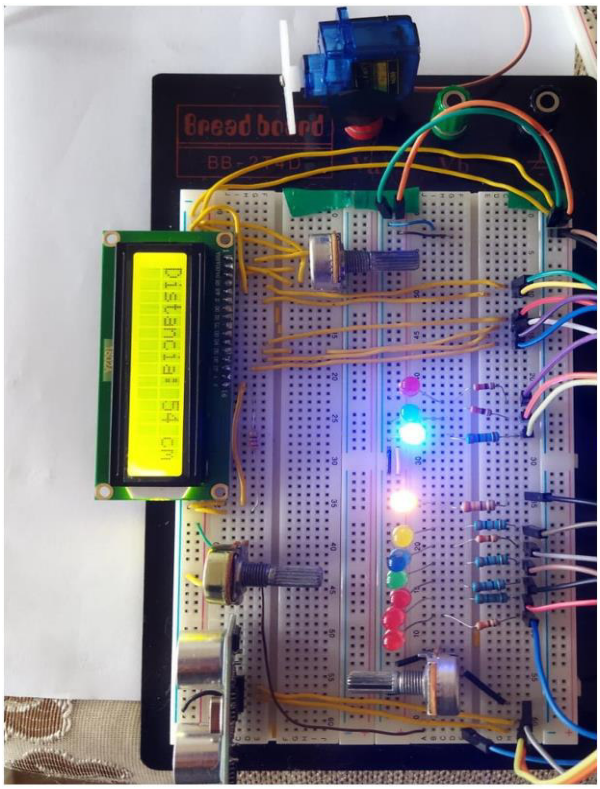


Figura 3.12 Prueba de periféricos sin FreeRTOS

Posterior a esto se utiliza el sistema operativo FreeRTOS con la misma programación y los mismos periféricos utilizados en la prueba anterior. Se muestra en la Figura 3.13 los periféricos funcionando con el sistema operativo FreeRTOS y se muestra los valores de la distancia, los leds encendidos al variar el potenciómetro, los leds de encendido y apagado y la posición del servomotor.

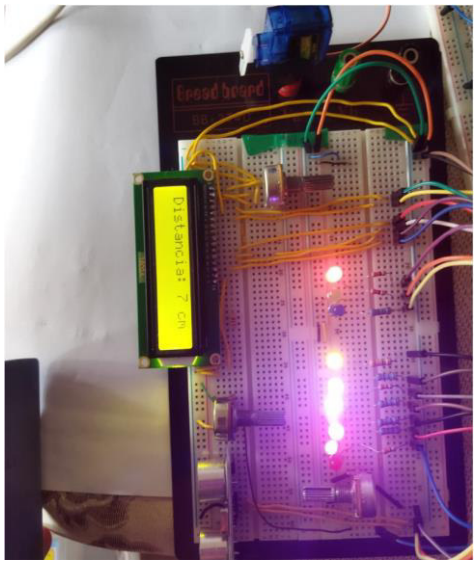


Figura 3.13 Prueba de periféricos con FreeRTOS

La Figura 3.14 se varios los parámetros de todos los periféricos y se muestra que todos los valores cambian al mismo tiempo cuando se los varia, evitando el problema de los retrasos de tiempo que se tuvo en la prueba anterior.

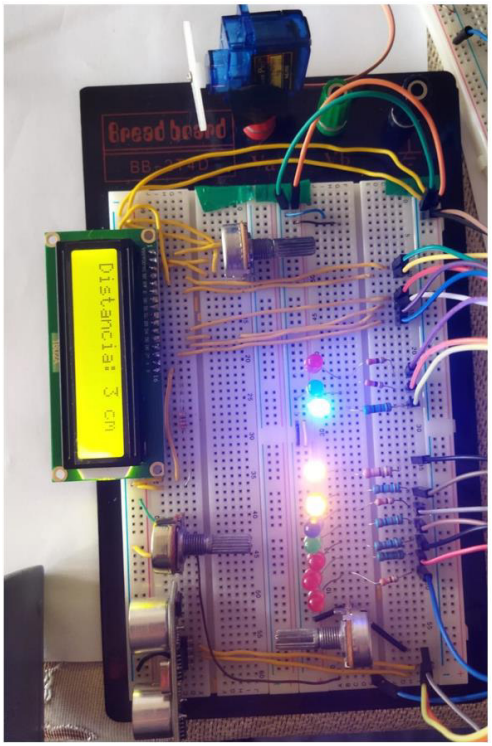


Figura 3.14 Prueba de periféricos con FreeRTOS

3.1.6 PRUEBA DEL PRIMER PROCESO AUTOMÁTICO

Para este proceso se utilizó las librerías de temporizadores para el momento de activarlo y desactivarlo. Al momento de presionar el pulsador se activa la electroválvula y entra en acción el temporizador TON, el cual da un retraso a la conexión, y posterior a esto empieza a funcionar, mientras que cuando se vuelva a presionar el pulsador entra en acción el segundo temporizador TOF, el que retrasará la desconexión del proceso.

El circuito presenta un cilindro de doble efecto conectada a una electroválvula, donde permite accionar de manera eléctrica y realizar el vaivén del actuador neumático, y se muestra en la Figura 3.15 el circuito para la prueba.

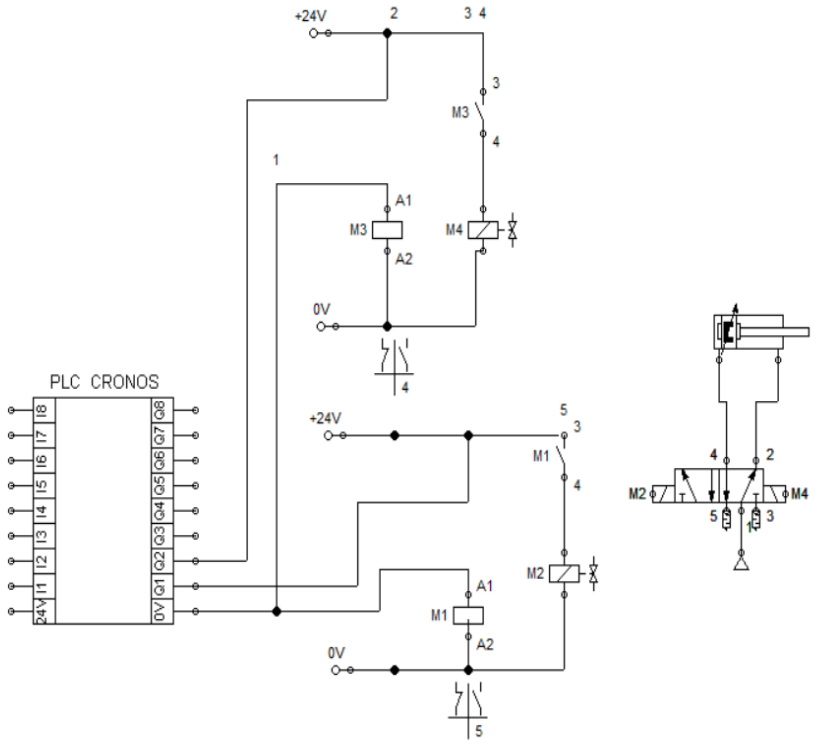


Figura 3.15 Circuito del actuador neumático

En la Figura 3.16 muestra el actuador neumático en estado inicial, donde se ha presionado el botón de marcha y está esperando que el temporizador termine de contar el tiempo, el cual se ha establecido en 2 segundos luego de lo cual energiza la electroválvula.

Figura 3.16 Prueba de encendido del actuador neumático

Una vez que completo el tiempo, se energiza la electroválvula, la que me permite el ingreso del aire al actuador neumático haciendo que su vástago empiece a salir. La Figura 3.17 muestra el momento en que mi electroválvula fue activada y permite que el vástago se salga.

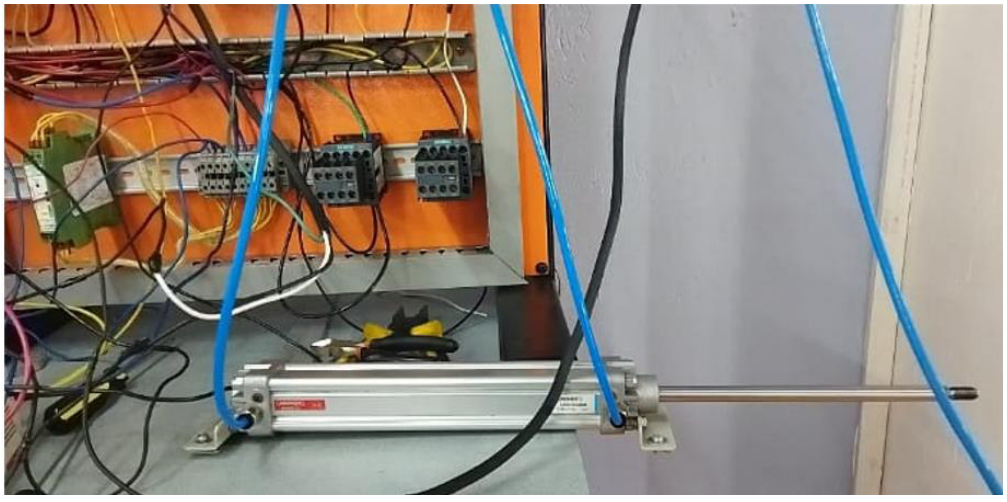


Figura 3.17 Prueba de encendido del actuador neumático

La Figura 3.18 muestra los resultados que se obtuvo al realizar esta prueba, donde se ve como la entrada fue activada y espera 2 segundos para que mi electroválvula entre en funcionamiento y empiece a realizar el vaivén con el actuador neumático.

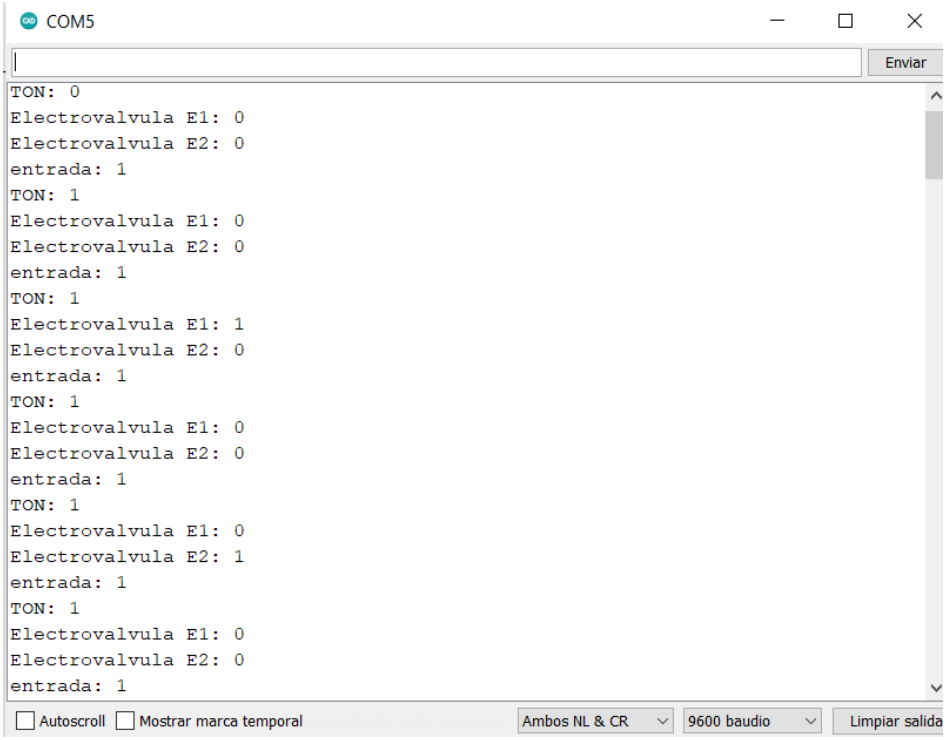


Figura 3.18 Datos de la prueba de vaivén del actuador neumático

3.1.7 PRUEBA DEL SEGUNDO PROCESO AUTOMÁTICO

La Figura 3.19 muestra el circuito completo de los procesos automáticos, donde se tiene la forma en la que se realiza la conexión de los diferentes dispositivos con el PLC.

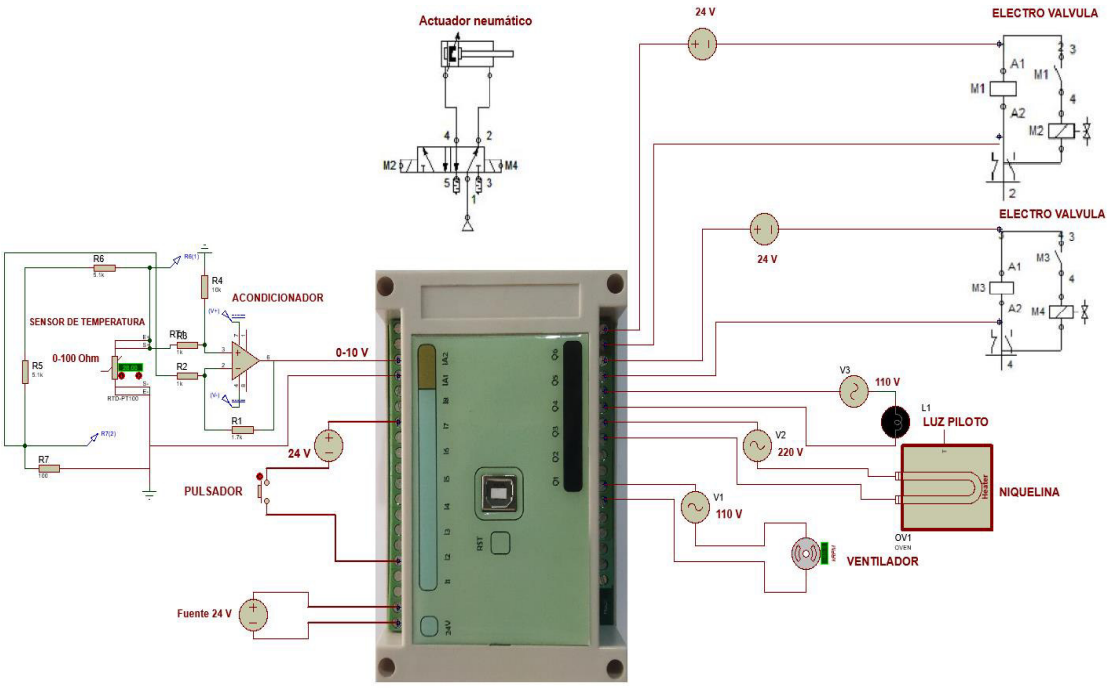


Figura 3.19 Circuito completo de los procesos automáticos

Para el segundo proceso se verifica el control de temperatura por histéresis, para lo cual se realizara un prueba de funcionamiento midiendo los voltajes y verificando que se enciendan las salidas tanto de la niquelina como de los ventiladores, para esto se utiliza la librería SCP la que me permite escalar los valores analógicos que llega desde el sensor tipo RTD, se utiliza la librería CTUD para aumentar un contador cada vez que el horno haga el control on/off con histéresis, y cuando llegue al valor establecido detener el proceso.

Para probar el funcionamiento del control de temperatura por histéresis, se toma un SetPoint de 40 grados con un rango de más menos 5 grados, como se muestra en la Figura 3.20, donde al bajar los 5 grados por debajo del SetPoint se activa la niquelina mientras que si aumenta los 5 grados por encima del SetPoint se activa los ventiladores.

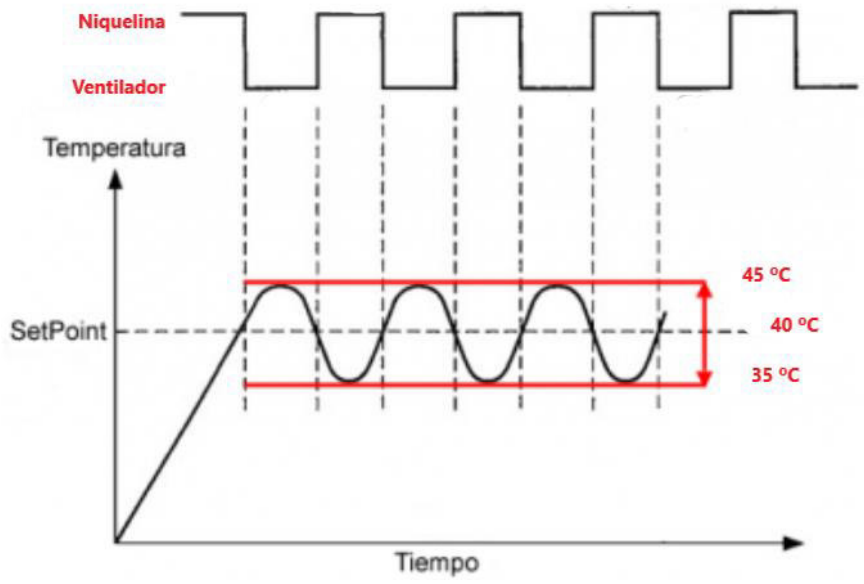


Figura 3.20 Gráfica del control ON/OFF con histéresis

La primera prueba se realizó a una temperatura baja para comprobar que la niquelina se enciende en estos rangos, en la Figura 3.21 muestra el voltaje que entrega el sensor a esta temperatura y como se observa la niquelina se activa para poder seguir incrementando la temperatura hasta el rango establecido, por lo que se verifica que el incremento de temperatura está funcionando correctamente.

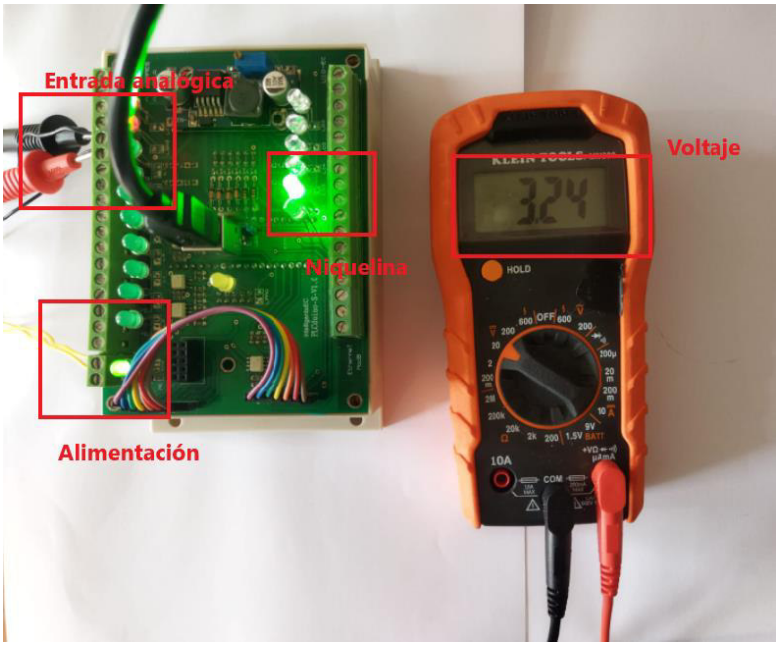


Figura 3.21 Prueba de funcionamiento a baja temperatura

La Figura 3.22 muestra los estados del ventilador y la niquelina a la temperatura de aproximadamente 35.6 grados centígrados y como se muestra en la Figura 3.21 se tiene un voltaje de 3.24 voltios que será el valor enviado por el sensor de temperatura al PLC.

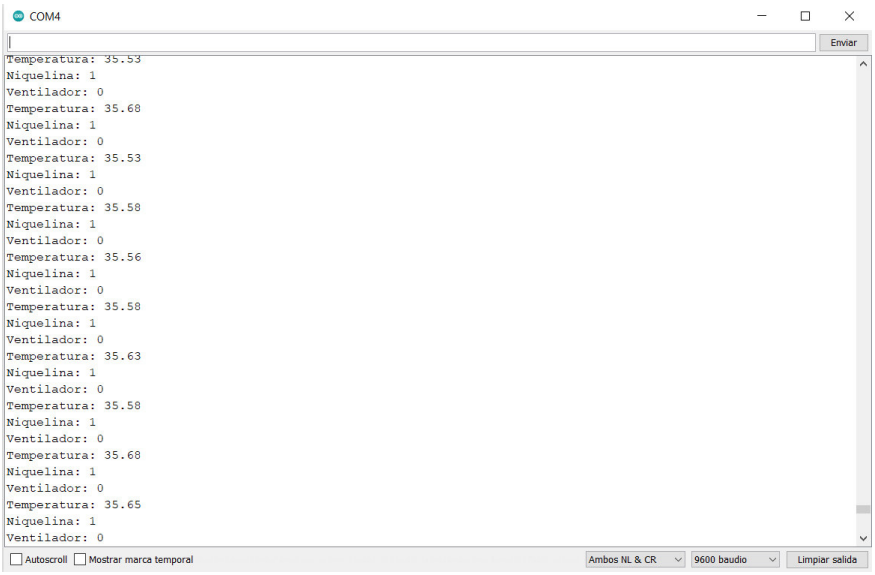


Figura 3.22 Datos de la prueba a baja temperatura

La segunda prueba se realizó a una temperatura alta para comprobar que los ventiladores se encienden en estos rangos, en la Figura 3.23 se muestra el voltaje que entrega el sensor a esta temperatura y como una vez alcanzado el límite superior se enciende el ventilador

para empezar a disminuir la temperatura, por lo que se verifica que los ventiladores que disminuyen la temperatura están funcionando correctamente.

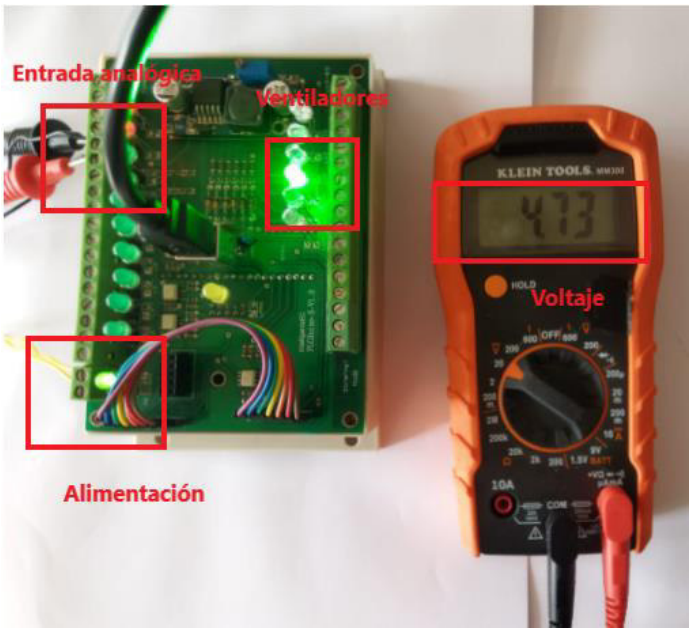


Figura 3.23 Prueba de funcionamiento alta temperatura

La Figura 3.24 muestra los estados del ventilador y la niquelina a la temperatura de aproximadamente 50.2 grados centígrados y como se muestra en la Figura 3.23 se tiene un voltaje de 4.73 voltios que será el valor enviado por el sensor de temperatura al PLC.

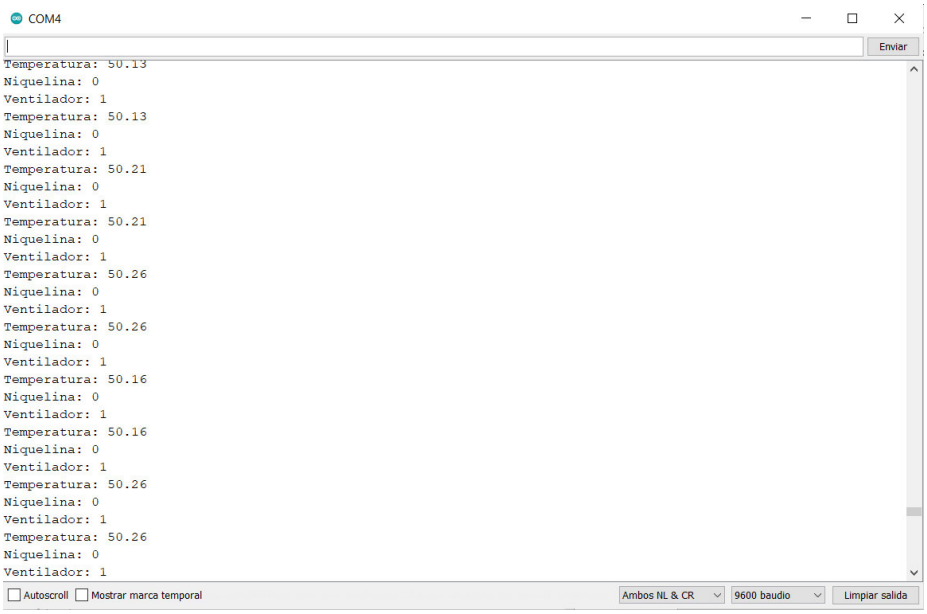


Figura 3.24 Datos de la prueba a alta temperatura

Una vez que se comprobó que la niquelina y los ventiladores están trabajando de manera correcta se realiza la prueba en el horno con una temperatura que está por debajo del valor

establecido. La Figura 3.25 muestra el horno con una luz indicadora que se encuentra encendida, lo que indica que la niquelina se encuentra activada y calentando el horno.

Figura 3.25 Luz indicadora del encendido de la niquelina

Cuando la temperatura llegue al valor máximo establecido la niquelina se apaga al mismo tiempo que su luz indicadora, y entra en funcionamiento los ventiladores, la Figura 3.26 muestra cómo se apagó la luz indicadora de la niquelina y se encendieron las luces indicadoras de los 3 ventiladores que tiene el horno. Los ventiladores seguirán funcionando hasta que la temperatura llegue a su valor mínimo establecido, donde se volverá a activar la niquelina.

Figura 3.26 Luz indicadora del encendido de los ventiladores

Se muestra en la Tabla 3.2 diferentes valores de temperatura tanto experimental como teóricos, los cuales se obtuvieron mediante el monitor serial de Arduino y de manera matemática para verificar que los valores que muestran en las Figuras 3.21 y 3.23 son correctos. Se puede verificar que los valores son muy similares ya que su error es relativamente bajo.

Tabla 3.2 Valores de voltaje y temperatura

VOLTAJE	**TEMPERATURA (Experimental)**	**TEMPERATURA (Teórico)**	**ERROR (%)**
2.54	31.19	31.2	0.35
3.14	37.22	36.9	0.859
3.72	42.93	42.6	0.77
4.04	47.49	46.2	2.79
4.36	50.15	49.55	1.21
4.72	51.92	51.80	0.23
4.99	55.84	55.5	0.61

3.1.8 PRUEBAS GLOBALES DE LOS DOS PROCESOS AUTOMÁTICOS

Una vez comprobado que los procesos son funcionales de manera individual, se realiza los mismos procesos, pero de manera conjunta y utilizando el sistema operativo FreeRTOS.

La primera prueba se realizará únicamente encendiendo el primer proceso donde la niquelina empezará a calentar el horno y el actuador neumático estará desactivado, comprobando que el primer proceso está funcionando de manera adecuada. La Figura 3.27 muestra como la luz indicadora de la niquelina se enciende mientras que el actuador neumático no se activa.

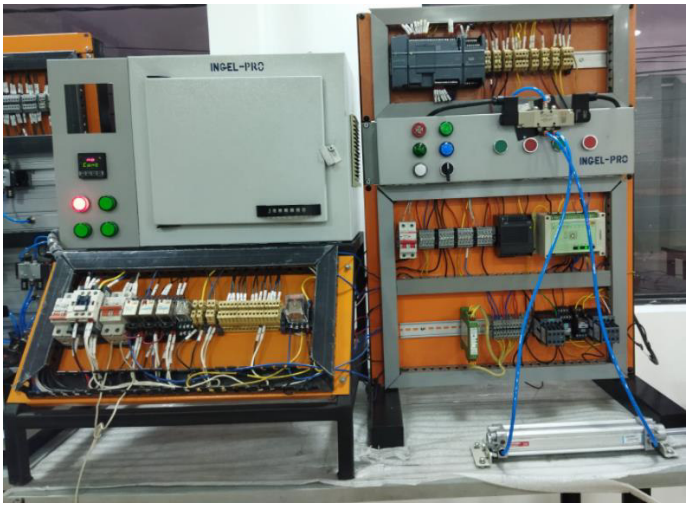


Figura 3.27 Encendido de la niquelina

La segunda prueba se realiza encendiendo el segundo proceso al accionar el pulsador el cual dará inicio al contador del tiempo de retardo y una vez acabado dicho tiempo empezará el vaivén del actuador neumático. La Figura 3.28 muestra como la niquelina sigue trabajando y la luz piloto del actuador neumático se encendió, lo que implica que el actuador neumático empezó a trabajar.

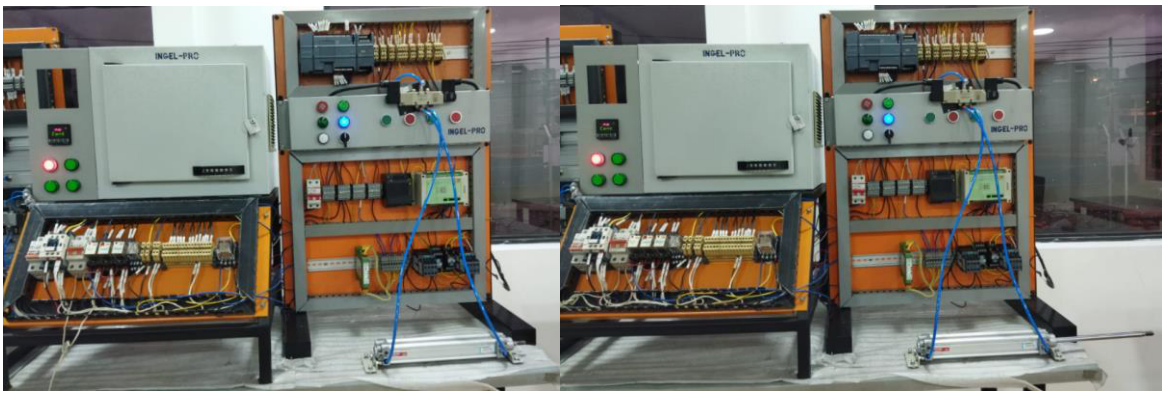


Figura 3.28 Encendido de ambos procesos

La Figura 3.29 muestra como una vez alcanzado la temperatura establecida la niquelina se desactiva y los ventiladores entran en funcionamiento, pero el vaivén del actuador neumático sigue trabajando, aunque el otro proceso presente cambios.

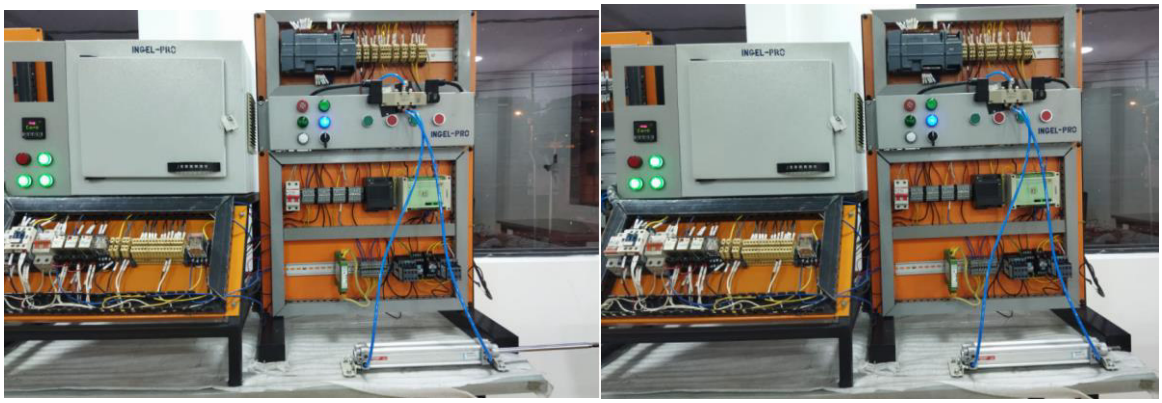


Figura 3.29 Encendido de ambos procesos

3.2 Conclusiones

- Una vez finalizadas las pruebas planteadas, se concluye que se ha cumplido con satisfacción los alcances y objetivos del presente trabajo de titulación.
- Luego de realizar la revisión bibliográfica, se concluye que en el país no hay fabricación nacional de PLCs con un sistema operativo FreeRTOS que permitan realizar la programación en la plataforma de Arduino IDE.
- Cuando se usa programación estructurada y el microcontrolador no tiene la velocidad suficiente para ejecutar el programa, va a existir problemas con retardos en procesamiento de información, por lo cual para este tipo de procesos es necesario el uso de un sistema operativo que realice las actividades como tareas en forma paralela.
- El uso de librerías de bloques de función predefinidos contribuye con la programación de procesos automáticos ya que permiten reducir las líneas de código y brindar herramientas para realizar programación de procesos al igual que otros lenguajes utilizados en la industria, por lo cual desarrollar este tipo de librerías para la plataforma de Arduino IDE permite familiarizar a las personas con los lenguajes cotidianos.
- Al realizar el proceso de instalación del sistema operativo FreeRTOS para el presente trabajo de integración curricular, se concluyó que para evitar posibles errores se instale la placa Generic STM32F1 series, ya que la librería de FreeRTOS únicamente está definida para esta placa, ya que para similares como la Generic STM32F103c series presenta errores al compilar.
- Para poder realizar el proceso de programación del sistema operativo lo más importante es como distribuir la memoria entre las tareas, por lo que se concluye que se debe programar las tareas en modo estático y con una pila adecuada para

cada tarea, con el fin de evitar gastos innecesarios de los recursos del microcontrolador.

- Al realizar pruebas, se realizó un código secuencial con múltiples tareas donde se generó errores al utilizar retardos, el cual generaba retrasos para recibir y enviar datos, pero al utilizar el mismo código en el sistema operativo se solucionó todos los errores aun utilizando los mismos retardos, por lo que se concluye que el uso del sistema operativo FreeRTOS es de utilidad cuando se maneja tiempos en las tareas.

3.3 Recomendaciones

- Se recomienda tener cuidado con la placa que se utiliza para compilar el PLC, nunca se debe utilizar placas similares o dañara la configuración de la STM32 y se tendría que extraer el microcontrolador del PLC y volver a realizar la configuración.
- Se recomienda el uso de tareas estáticas, ya que las dinámicas presentan algunos problemas como fragmentación de memoria y generar errores en el llamado de tareas.
- Se recomienda hacer pruebas de pila para las tareas del sistema operativo, con el fin de utilizar el menor número posible de recursos del microcontrolador.
- Para futuras implementaciones se recomienda la creación de más librerías para trabajar de manera más aproximada a los lenguajes comúnmente usados en la industria.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] Quality RTOS & Embedded Software, «RTOS - Free professionally developed and robust real time operating system for small embedded systems development» Febrero 2009. [En línea]. Available: http://www.openrtos.net/RTOS.html (accedido 18 de noviembre de 2021).

[2] E. Rincón Cruz, «FreeRTOS-Wiki» Agosto 2017. [En línea]. Available: http://www.coffeebrain.org/wiki/index.php?title=FreeRTOS#Anatom.C3.ADa_de_un_proyecto_con_FreeRTOS (accedido 18 de noviembre de 2021).

[3] D Rojas, «Desarrollo de una biblioteca bajo el Paradigma Multiagente para Sistemas Embebidos (MAES) compatible con la NanoMind A3200 y el kernel FreeRTOS» Tecnológico de Costa Rica, Cartago, Abril 2021. Available: https://repositoriotec.tec.ac.cr/bitstream/handle/2238/13300/TFG_Daniel_Rojas_Marin.pdf?sequence=1&isAllowed=y (accedido 18 de noviembre de 2021).

[4] J. Rodríguez, «Tareas estáticas en FreeRTOS para Arduino – Electrónica, PCBs y Sistemas embebidos» Enero 2021. [En línea]. Available: http://fjrg76.com/2021/01/05/tareas-estaticas-en-freertos-para-arduino/ (accedido 25 de noviembre de 2021).

[5] *Amazon Web Services,* «FreeRTOS, sistema operativo con funcionamiento en tiempo real para microcontroladores - AWS» Marzo 2018 [En línea]. Available: https://aws.amazon.com/es/freertos/ (accedido 18 de noviembre de 2021).

[6] T. Carrión, «Diseño y construcción de un controlador lógico programable, basado en una plataforma de 32 bits programable en Arduino IDE» Escuela Politécnica Nacional, Quito, Ecuador, Noviembre 2019.

[7] V Suarez, «Temporizador de retardo a la desconexion TOF» 20 Noviembre 2007. [En línea]. Available: http://www.isa.uniovi.es/~vsuarez/ii/CursoOnline/7dtemporizador%20TOF.htm (accedido 18 de noviembre de 2021).

[8] V Suarez «Contador ascendente CTU» Noviembre 2007 [En línea]. Available: http://isa.uniovi.es/~vsuarez/ii/CursoOnline/8bcontadores%20CTU.htm (accedido 18 de noviembre de 2021).

[9] D. García, «RSLogix 500 Escalado entrada analógica Función SCP» Febrero 2011 [En línea]. Available: https://www.infoplc.net/descargas/193-rockwell/software-programacion/306-rslogix-500-escalado-entrada-analogica-funcion-scp (accedido 18 de noviembre de 2021).

[10] WIKA, «Resistance thermometer For additional thermowell Model T» Enero 2009, [En línea]. Available: https://kz.wika.com/upload/DS_TE6002_en_co_6991.pdf (accedido 17 de enero de 2022).

[11] Directindustry, «CP-20 - Ventilador axial by COPPUS | DirectIndustry» [En línea]. Available: https://www.directindustry.es/prod/coppus/product-13998-591246.html (accedido 17 de enero de 2022).

[12] A. Sánchez «T-ESPEL-EMI-0255-P.pdf». Agosto 2016. [En línea]. Available: http://repositorio.espe.edu.ec/bitstream/21000/8300/2/T-ESPEL-EMI-0255-P.pdf (accedido 17 de enero de 2022).

[13] Parker, «ESTechP1D.pdf». Enero 2021. [En línea]. Disponible en: https://www.parker.com/parkerimages/euro_pneumatic/cat/Es/ESTechP1D.pdf

[14] Rotork, «pub124-003-04-0519.pdf». Enero 2021. [En línea]. Available: https://www.rotork.com/uploads/documents-versions/44547/1/pub124-003-04-0519.pdf

[15] Mouser electronics, «Apem Interruptores con pulsador Fichas técnicas – Mouser Ecuador» [En línea]. Available: https://www.mouser.ec/c/ds/electromechanical/switches/pushbutton-switches/?m=Apem&data%20sheet%20link=%2Fdatasheet%2F2%2F26%2Fpusbutton-switches-serie-IM-1291428.pdf (accedido 20 de enero de 2022).

[16] R. Aswinth, «Primeros pasos con la placa de desarrollo STM32F103C8T6 STM32 (Blue Pill) con Arduino IDE: LED parpadeante» Agosto 2018. [En línea]. Available: https://circuitdigest.com/microcontroller-projects/getting-started-with-stm32-blue-pill-development-board-stm32f103c8-using-arduino-ide (accedido 17 de enero de 2022).

[17] J Carvajal Godínez, «Desarrollo tecnológico de un sistema de adquisición de datos ambientales para su uso en proyectos de investigación científica: Arquitectura abierta CRTecMote» Instituto tecnológico de Costa Rica, 2009-2010. [En línea]. Available: https://repositoriotec.tec.ac.cr/bitstream/handle/2238/5787/desarrollo-tecnol%C3%B3gico-sistema-adquisici%C3%B3n-datos.pdf?sequence=1&isAllowed=y (accedido 17 de enero de 2022).

[18] F. Rus «FreeRTOS - ESP32 - — Guía de programación de ESP-IDF documentación más reciente» Octubre 2018. [En línea]. Available: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos.html (accedido 17 de enero de 2022).

5 ANEXOS

ANEXO I. Manual de instalación de tarjeta STM32F103C

ANEXOII. Hojas de datos

ANEXO I

MANUAL DE INSTALACIÓN DE TARJETA STM32F103C

I.1. INTRODUCCIÓN

CRONOS es un controlador lógico programable desarrollado por la Escuela Politécnica Nacional que enfrenta algunos desafíos como trabajar con varios procesos a la vez, por lo cual se implementó un sistema operativo que permita solucionar los problemas de que se presentan en un desarrollo secuencial, permitiéndole trabajar adecuadamente con procesos industriales.

I.1.1 OBJETIVO

El objetivo del presente manual de usuario es mostrar la manera de instalar y configurar la tarjeta STM32, e implementar la librería que permite usar el sistema operativo FreeRTOS, con el fin de poder utilizar y programar el PLC CRONOS en cualquier computador.

I.2. CARACTERÍSTICAS

Este apartado tendrá dos etapas, la primera muestra la configuración e instalación de la tarjeta STM32F103c en la plataforma de arduino IDE y en la segunda se muestra la instalación de la librería del sistema operativo FreeRTOS.

I.3 GUÍA PARA INSTALACIÓN DE TARJETA STM32F103C

En esta sección se detalla el proceso de instalación de la tarjeta STM32F1 para poder para poder programar el controlador lógico programable CRONOS.

I.3.1. Instalación de tarjeta STM32F1xx

El primer paso para poder programar, es instalar la tarjeta stm32F103C en la plataforma de Arduino, para lo cual se abre un nuevo programa de Arduino y como se muestra en la Figura I.1, y se selecciona gestor de tarjetas.

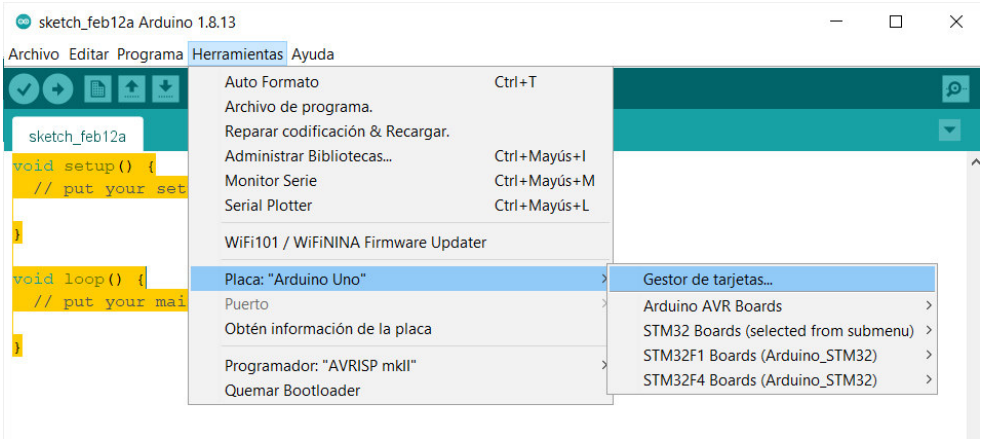


Figura I.1 Gestor de tarjetas

Al ingresa al gestor de tarjetas como se muestra en la Figura I.2 se muestran varias opciones de tarjetas de la familia STM, para este caso se instala la segunda opción STM32F1xx/GD32F1xx boards.

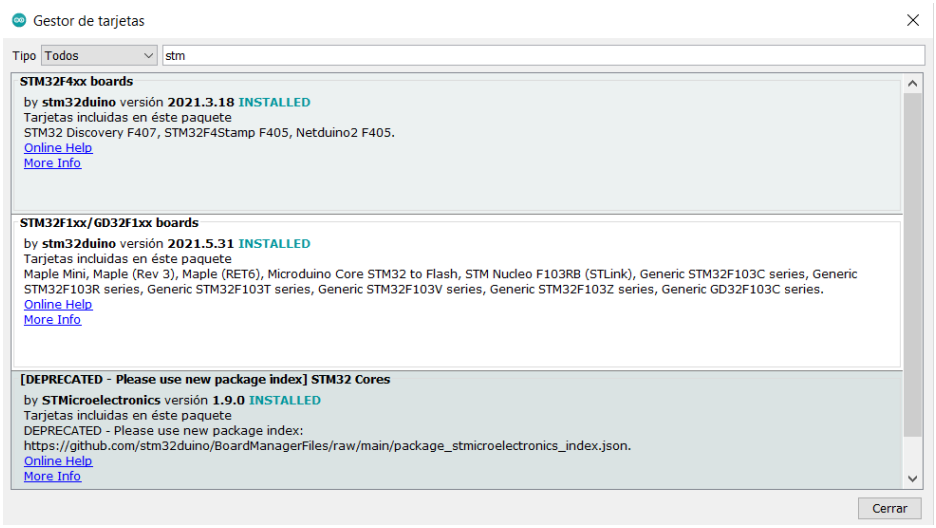


Figura I.2 Instalación de tarjeta STM32F1xx

Una vez instalada, se debe dirigir a herramientas y seleccionar la tarjeta instalada, a continuación, se realiza las configuraciones que se muestra en la Figura I.3.

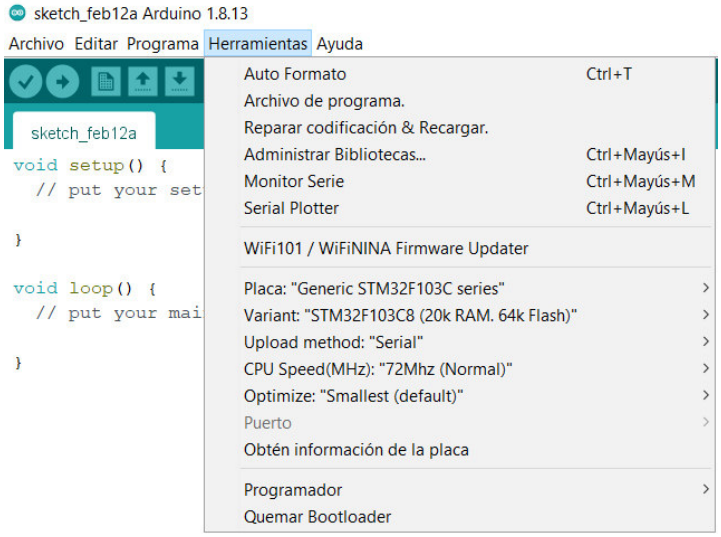


Figura I.3 Configuración de la tarjeta STM32F103C

I.3.2 Programación serial

Como segundo paso se debe conectar la tarjeta STM32 y la FTD1, el que permite la programación de la tarjeta de forma serial, tal y como se muestra en la Figura I.4 donde se observa los pines en los que se debe conectar tanto la FTD1 y la tarjeta STM.

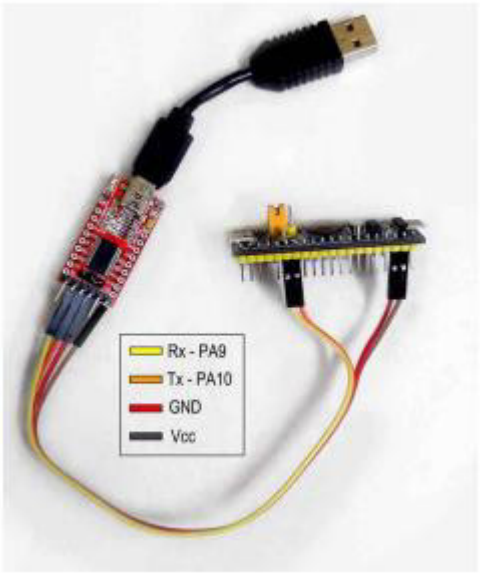


Figura I.4 Conexión de la tarjeta STM32F103C

Antes de conectar al computador, se debe cambiar los jumpers de la parte superior de la tarjeta, tal y como se muestra en la Figura I.5.

Figura I.5 Jumpers

Como tercer paso se debe descargar los archivos necesarios que serán utilizados para la configuración, ingresando al link https://www.st.com/en/development-tools/stm32cubeprog.html, se encuentra el instalador del flasher de STM que se utiliza para cargar un archivo.bin, el que será descargado en la siguiente dirección: https://github.com/rogerclarkmelbourne/STM32duino-bootloader/blob/master/binaries/.

Una vez instalado el programa de STM, se lo inicia y se cambia su modo de configuración a UART, tal y como se muestra en la Figura I.6.

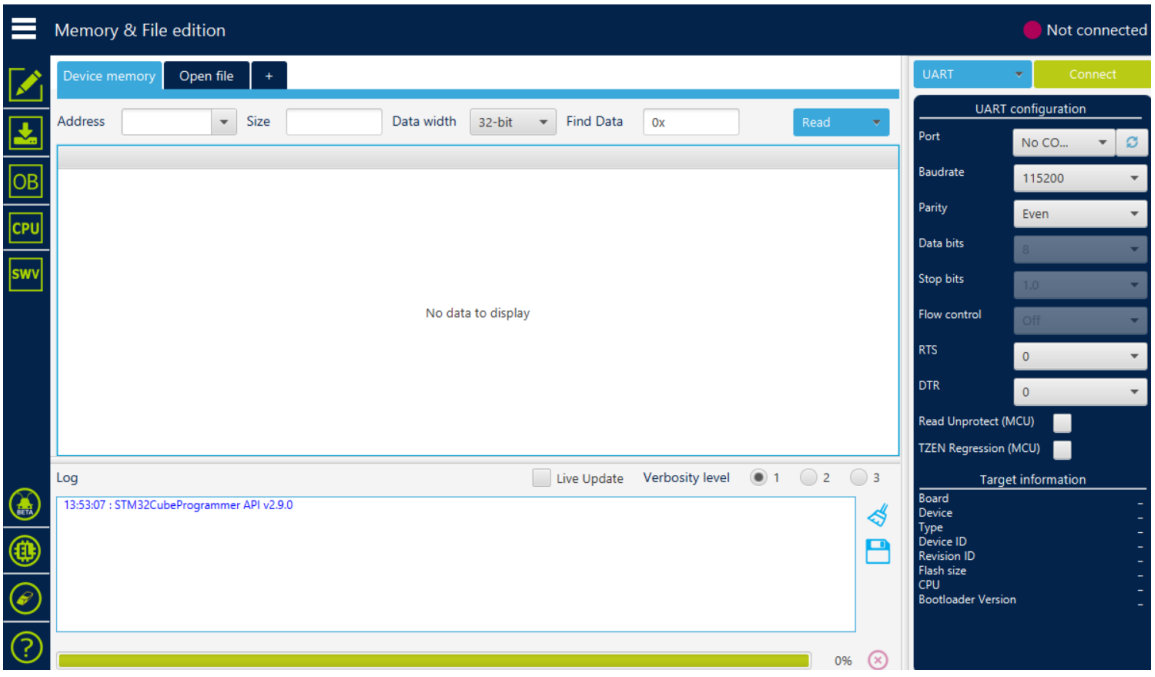


Figura I.6 Programa de STM

Al conectar la tarjeta como se muestra en la Figura I.7, aparece el puerto de la computadora, por lo que está listo para conectar, una vez aplastado el botón de conectar, debe aparecer una señal en verde que diga conectado.

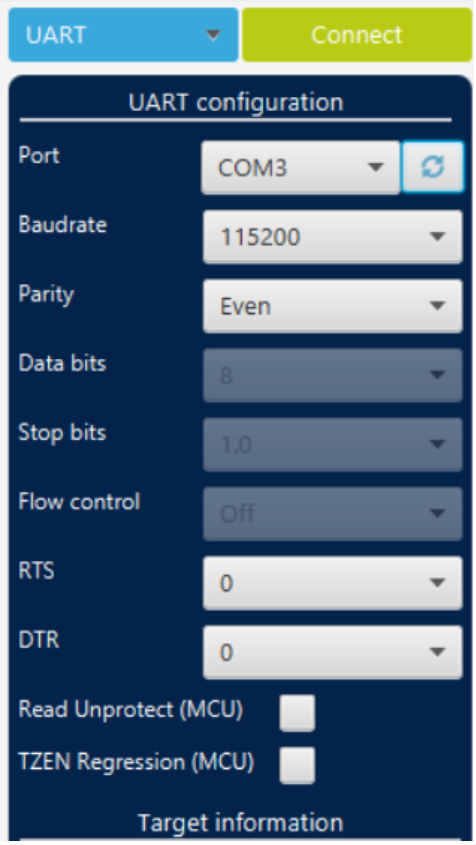


Figura I.7 Configuración UART

Cuando la tarjeta ya se encuentre conectada, se debe dar click en erasing & programming para cargar el archivo .bin. En la Figura I.8 se muestra la luz indicadora que señala que se encuentra conectada la tarjeta y donde se encuentra erasing & programming para cargar el archivo .bin.

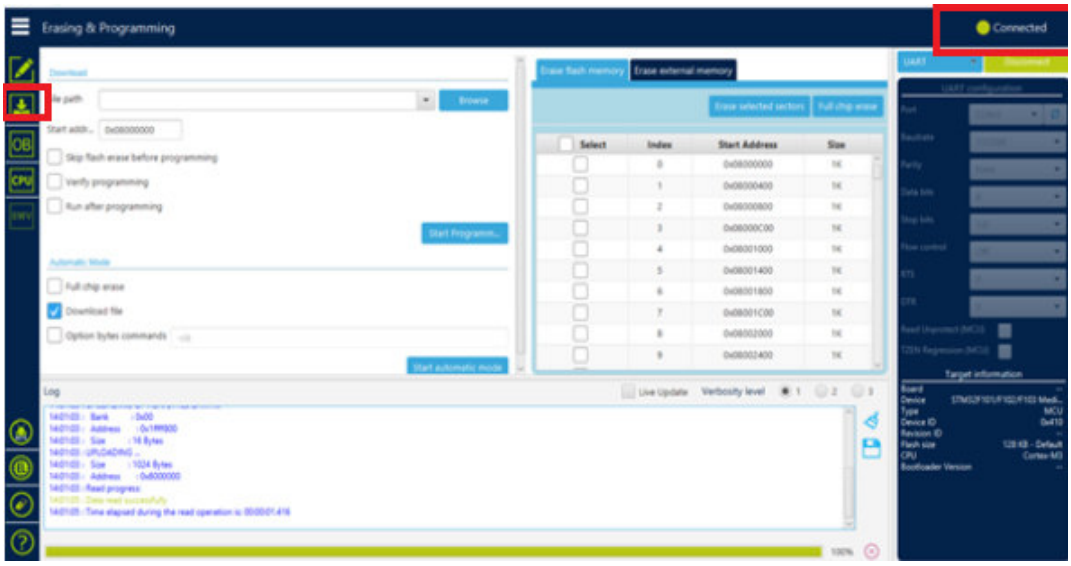


Figura I.8 Erasing & programming

Se debe hacer click en Browse y buscar donde se haya guardado el archivo .bin que se descargó en el enlace, y cargar el archivo generic_boot20_pc13.bin.

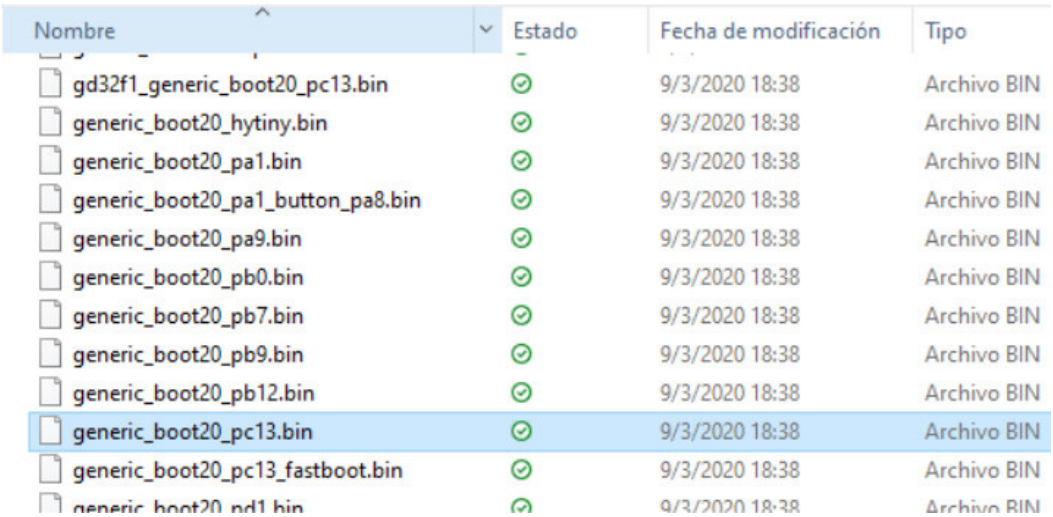


Figura I.9 Archivo .bin

Una vez cargado el programa, se carga los archivos a la tarjeta aplastando Start Programming.

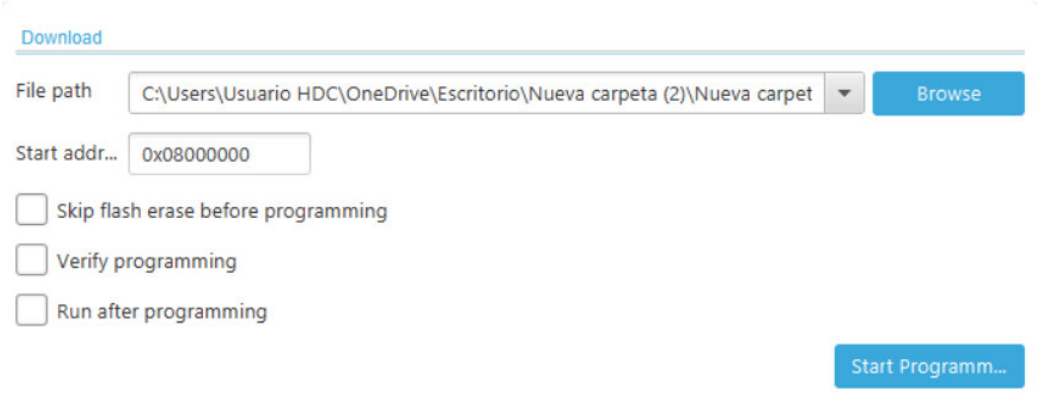


Figura I.10 Cargar programa en la tarjeta

Una vez acabado de cargar los archivos, se debe cambiar nuevamente la posición de los jumpers, ambos deben estar en cero y se debe realizar este proceso antes de desconectar la tarjeta.

I.3.3 Sistema operativo FreeRTOS

Para la instalación de la librería del sistema operativo FreeRTOS, hay que dirigirse al gestor de librerías e instalar la librería STM32duino FreeRTOS. La Figura I.11 muestra la librería que debe ser instalada.

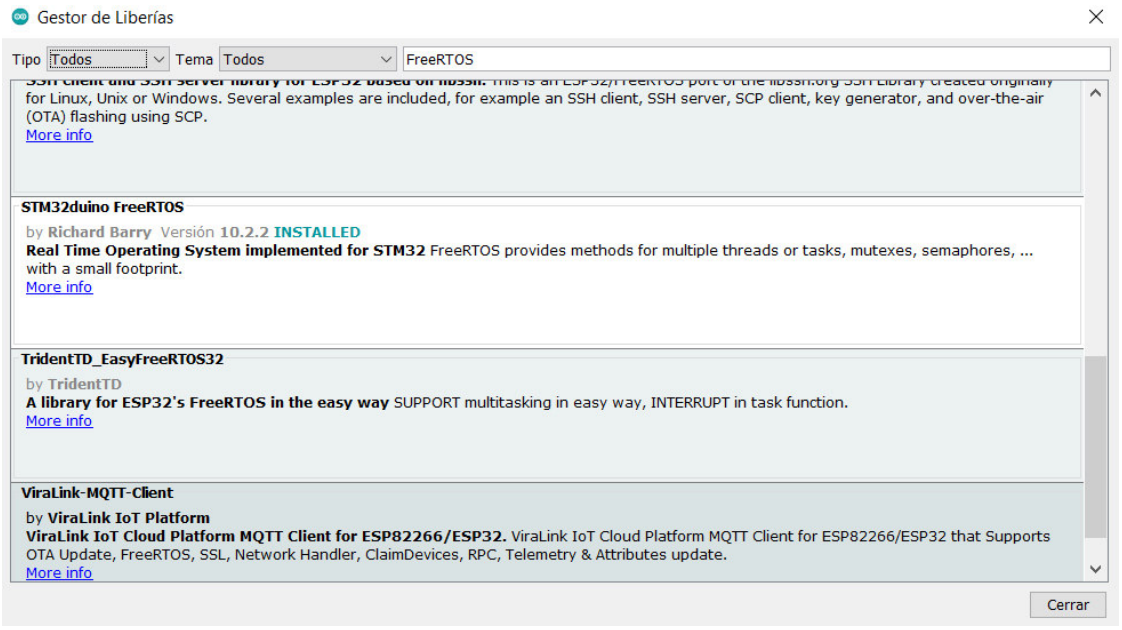


Figura I.11 Libreria del sistema operativo FreeRTOS

ANEXO II

HOJA DE DATOS

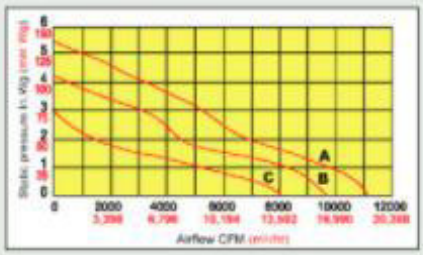
COPPUS
CP-20

Air- or steam turbine-driven blower/exhauster

DESCRIPTION

This powerful fan is designed for fast and thorough degassing, ventilating or cooling of large process vessels such as columns, towers, reactors, scrubbers, furnaces, and storage tanks.

MODEL/SPECIFICATIONS

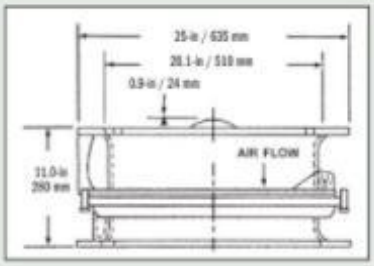


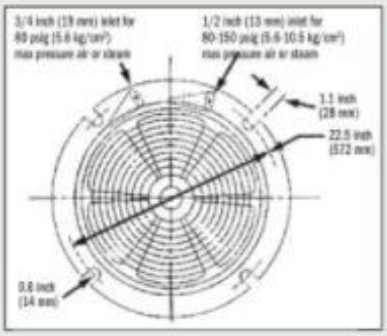
- A = 80 psig (5, 6 kg/cm2) at large nozzle or 150 psig (10,6 kg/cm2) at small
- B = 60 psig (4, 2 kg/cm2) at large nozzle or 115 psig (8, 1 kg/cm2) at small
- C = 40 psig (2, 8 kg/cm2) at large nozzle or 80 psig (5, 6 kg/cm2) at small

FEATURES / ADVANTAGES

- Delivers air flow up to 11,200 cfm (19,029 m3/hr)
- Can be used as blower or exhauster
- Fits 20 in (508 mm) API tank opening
- Cast aluminum housing and fan blade
- Stainless steel turbine buckets
- Separate stainless steel nozzles for high- or low-pressure operation
- Fan assembly shaft rotates on permanently sealed ball bearings
- Stationery expansion nozzles

DIMENSIONS





TECHNICAL DATA

AIR AND STEAM DRIVEN		
CP-20	80 psig	108 dBA
CP-20	60 psig	107 dBA
CP-20	40 psig	105 dBA

High-pressure inlet equals small nozzle

- 1/2 in NPT connection

Low-pressure inlet equals large nozzle

- 3/4 in NPT connection

STEAM AND AIR CONSUMPTION				
STEAM/AIR PRESSURE psig kg/cm²	STEAM CONSUMPTION lbs/hr / kg/hr		AIR CONSUMPTION scfm / m³/hr	
	SMALL NOZZLE	LARGE NOZZLE	SMALL NOZZLE	LARGE NOZZLE
150 / 10.6	640 / 209		220 / 178	
115 / 8.1	510 / 231		178 / 302	
80 / 5.6	380 / 172	740 / 336	128 / 217	250 / 425
60 / 4.2		590 / 268		194 / 330
40 / 2.8		440 / 200		142 / 241

Resistance thermometer
For additional thermowell
Model TR10-B

for further approvals
see page 15

Applications

- Machine building, plant and vessel construction
- Energy and power plant technology
- Chemical industry
- Food and beverage industry
- Sanitary, heating and air-conditioning technology

Special features

- Sensor ranges from -196 ... +600 °C [-320 ... +1,112 °F]
- For mounting in all standard thermowell designs
- Spring-loaded measuring insert (replaceable)
- Pt100 or Pt1000 sensors
- Explosion-protected versions are available for many approval types (see page 2)

Fig. left: Model TR10-B with BSZ connection head
Fig. right: Model TR10-B with 1/4000 connection head

Description

Resistance thermometers in this series can be combined with a large number of thermowell designs. Operation without thermowell is only recommended in certain applications.

A wide variety of possible combinations of Pt100 or Pt1000 sensor, connection head, insertion length, neck length, connection to thermowell etc. are available for the thermometers; suitable for any thermowell dimension and any application.

A large number of different explosion-protected approvals are available for the TR10-B.

Optionally we can fit analogue or digital transmitters from the WIKA range into the connection head of the TR10-B.

Data sheets showing similar products:
Thermocouple for additional thermowell, model TC10-B; see data sheet TE 65.02
Threaded resistance thermometer, model TR10-C; see data sheet TE 60.03
Threaded thermocouple, model TC10-C; see data sheet TE 65.03

WIKA
Part of your business

Fuerzas de los cilindros de doble efecto

Diám. cil. vástago mm	Carrera	Área piston cm²	Max. fuerza teórica en N (bar) 1,0	2,0	3,0	4,0	5,0	6,0	7,0	8,0	9,0	10,0
32/12	+	8,0	80	161	241	322	402	**483**	563	643	724	804
	-	6,9	69	138	207	276	346	**415**	484	553	622	691
40/16	+	12,6	126	251	377	503	628	**754**	880	1005	1131	1257
	-	10,6	106	212	318	424	530	**636**	742	848	954	1060
50/20	+	19,6	196	393	589	785	982	**1178**	1374	1571	1767	1963
	-	16,5	165	330	495	660	825	**990**	1155	1319	1484	1649
63/20	+	31,2	312	623	935	1247	1559	**1870**	2182	2494	2806	3117
	-	28,0	280	561	841	1121	1402	**1682**	1962	2242	2523	2803
80/25	+	50,3	503	1005	1508	2011	2513	**3016**	3519	4021	4524	5027
	-	45,4	454	907	1361	1814	2268	**2721**	3175	3629	4082	4536
100/25	+	78,5	785	1571	2356	3142	3927	**4712**	5498	6283	7069	7854
	-	73,6	736	1473	2209	2945	3682	**4418**	5154	5890	6627	7363
125/32	+	122,7	1227	2454	3682	4909	6136	**7363**	8590	9817	11045	12272
	-	114,7	1147	2294	3440	4587	5734	**6881**	8027	9174	10321	11468

+ = Carrera de salida
- = Carrera de entrada

¡Atención!
Seleccionar una fuerza teórica 50-100% mas grande que la fuerza requerida

Datos generales: P1D

Cilindro, designación	Cilindro diá. mm	área cm²	Vástago diá. mm	área cm²	rosca	Tramo-amort. mm	Consumo-aire[2] libros	Rosca conexión	Dim. manguera. Flexible Porting mm
P1D-•032••-X	32	8,0	12	1,1	M10x1,25	17	0,105	G1/8	4 ó 6.
P1D-•040••-X	40	12,6	16	2,0	M12x1,25	19	0,162	G1/4	4 ó 6.
P1D-•050••-X	50	19,6	20	3,1	M16x1,5	20	0,253	G1/4	8 ó 10.
P1D-•063••-X	63	31,2	20	3,1	M16x1,5	23	0,414	G3/8	8 ó 10.
P1D-•080••-X	80	50,3	25	4,9	M20x1,5	23	0,669	G3/8	-
P1D-•100••-X	100	78,5	25	4,9	M20x1,5	27	1,043	G1/2	-
P1D-•125••-X	125	122,7	32	8,0	M27x2	30	1,662	G1/2	-

Masa total incluyendo partes móviles

Cilindro, designación	Masa total (kg) con carrera de 0 mm Standard	Tie-Rod	Clean/Flex	Suplemento masa (kg) para cilindro con bloqueo vástago todos los variantes	Masa total (kg) Complemento por 10 mm carrera Standard	Tie-Rod	Clean/Flex
P1D-•032••-X	0,55	0,54	0,60	0,31	0,023	0,022	0,047
P1D-•040••-X	0,80	0,79	0,88	0,44	0,033	0,030	0,063
P1D-•050••-X	1,20	1,20	1,32	0,61	0,048	0,048	0,094
P1D-•063••-X	1,73	1,73	1,86	1,25	0,051	0,051	0,101
P1D-•080••-X	2,45	2,47	2,63	2,45	0,075	0,079	0,142
P1D-•100••-X	4,00	4,00	4,22	3,72	0,084	0,084	0,168
P1D-•125••-X	6,87	6,73	7,01	6,07	0,138	0,129	0,248

Masa de las partes moviles (para el cálculo de la amortiguación)

Cilindro, designación	Masa piezas móviles (kg) con 0 mm carrera Todas las variantes	Complemento por Carrera 10 mm Todas las variantes
P1D-•032••-X	0,13	0,009
P1D-•040••-X	0,24	0,016
P1D-•050••-X	0,42	0,025
P1D-•063••-X	0,50	0,025
P1D-•080••-X	0,90	0,039
P1D-•100••-X	1,10	0,039
P1D-•125••-X	2,34	0,063

1) Carrera
2) Consumo de aire libre por 10 mm de carrera para doble embolada a 6 bares

IM series

Sealed snap-action pushbutton switches • bushing Ø 12 mm • momentary

DISTINCTIVE FEATURES

Snap-action : tactile feedback with audible click
High current/voltage rating
Sealed to IP67
Flat round actuator for optional marking
NO+NC

ENVIRONMENTAL SPECIFICATIONS

- Front panel sealing : IP67 according to IEC 60529
- Shock resistance : 50 g - 11 ms according to IEC 68-2-27
- Salt spray : IEC 512-6, test 11f
- Operating temperature : -40 °C to +85 °C (-40 °F to +185 °F)
- Storage temperature : -40 °C to +85 °C (-40 °F to +185 °F)

ELECTRICAL SPECIFICATIONS

- Max. current/voltage rating with resistive load : 3 A 28 VDC
- Initial contact resistance : 100 mΩ max.
- Insulation resistance : 1 GΩ min. at 500 VDC
- Dielectric strength : 500 Vrms between terminals
- Electrical life at full load : 25,000 cycles

GENERAL SPECIFICATIONS

- Panel thickness : 1.5 mm (.059) to 10 mm (.394)
- Total travel : 1.7 mm (.067) ± 0.3 mm
- Low level or mechanical life : 1,000,000 cycles
- Torque (applied to nut) : 1 Nm max. with metal nut U166
  1.5 Nm max. with plastic nut U4248

MATERIALS

- Case : polyamide 4/6
- Actuator : polyamide 6/6
- Bushing/bezel : polyamide 6/6
- Contacts : silver, gold plated
- Terminal seal : epoxy

Acondicionador de señales de entrada RTD

DRF-RTD

- **Elemento RTD de platino (Pt) de 100 Ω, curva de 0,00385**
- **Configuración de 2 o 3 hilos**
- **Precisión de 0,2%**
- **Compensación de resistencia del hilo de hasta 10 Ω**
- **Protección exclusiva contra roturas**
- **Tiempo de respuesta de < 250 mseg.**
- **Aislamiento galvánico entre la entrada, la salida y la alimentación**

Los acondicionadores de señal DRF-RTD RTD admiten RTD de platino de 2 o 3 hilos como entrada y proporcionan una salida aislada de 0 a 10 Vcc o de 4 a 20 mA. Los modelos están disponibles con tres opciones de alimentación diferentes: 24 Vcc, 120 Vca y 240 Vca.

Los DRF-RTD son idóneos para aplicaciones industriales. Todos los modelos se montan en un riel DIN estándar de 35 mm y ofrecen un aislamiento galvánico entre la entrada, la salida y la alimentación de hasta 3.500 Veff (en modelos específicos). El tiempo de respuesta del módulo es de 250 mseg. o inferior.

El modelo DRF-RTD-24VDC-0/100C-0/10 se muestra en un tamaño superior al real.

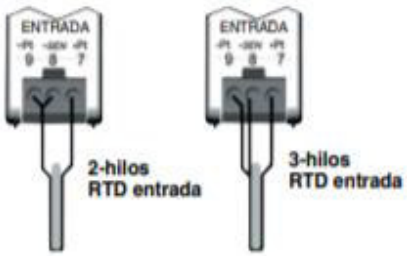
Especificaciones

RTD: 2 o 3 hilos 100Ω platino RTD, Ω=0,00385
Precisión: <0,2% escala completa
Linealidad: <0,1% escala completa
Deriva térmica: <250 ppm/°C típica
Tiempo de respuesta: <250 mseg. (90% de señal)
RTD Excitación: 1 Vcc
Impedancia de entrada: Medida con un puente de Wheatstone. Puente a positivo mediante resistencia de 100Ω, Puente a negativo mediante resistencia de 10KΩ.

Tabla de rangos de entrada

Código del rango:	Rango
-25/75 C	-25 a 75 °C
-50/150 C	-50 a 150 °C
0/100 C	0 a 100 °C
0/200 C	0 a 200 °C
0/300 C	0 a 300 °C
0/450 C	0 a 450 °C
0/600 C	0 a 600 °C

**Los rangos personalizados pueden obtenerse ajustando los potenciómetros integrados cero e intervalo. El rango mínimo es de 0 a 50 °C, el rango máximo es de 0 a 600 °C (32 a 1.112 °F).*



Para hacer su pedido, *visite es.omega.com/drf_series para consultar precios y detalles*

N.° de modelo	Descripción
DRF-RTD-(*)-(**)-(***)	Acondicionador de señal para RTD de 100 Ω Pt

** Especificar la alimentación, **"24 Vcc"** para alimentación de 24 Vcc, **"115 Vca"** para alimentación de 115 Vca o **"230 Vca"** para alimentación de 230 Vca*

*** Especificar código de rango a partir de la tabla de rangos de entrada*

**** Especificar salida, **"4/20"** para salidas de 4 a 20 mA o **"0/10"** para salidas de 0 a 10 Vcc*

***Ejemplo de pedido: DRF-RTD-24VDC-0/100C-0/10,** acondicionador de señal para RTD con rango de entrada de 0 a 100 °C, salida de 0 a 10 Vcc y alimentación de 24 Vcc.*