# LABVIEW

## UN ENFOQUE PRÁCTICO PARA PRINCIPIANTES

2023


**Ivaldo Torres Chávez**
**Diego Armando Mejía Bugallo**
**Wilson Andrés Suárez Villamizar**



UNIVERSIDAD DE PAMPLONA
SELLO EDITORIAL

# **LABVIEW** UN ENFOQUE PRÁCTICO PARA PRINCIPIANTES

# LABVIEW
# UN ENFOQUE PRÁCTICO PARA PRINCIPIANTES

Ivaldo Torres Chávez
Diego Armando Mejía Bugallo
Wilson Andrés Suárez Villamizar

UNIVERSIDAD DE PAMPLONA
Vicerrectoría de INVESTIGACIONES UNIPAMPLONA
Formando líderes para la construcción de un nuevo país en paz

*Labview un enfoque práctico para principiantes* / Ivaldo Torres Chávez, Diego Armando Mejía Bugallo, Wilson Andrés Suárez Villamizar -- Pamplona: Universidad de Pamplona. 2023.

218 p. ; 21,59 cm x 27,94 cm.
ISBN: 978-628-7656-04-8


Sede Principal Pamplona, Km 1 Vía Bucaramanga-
Ciudad Universitaria. Norte de Santander, Colombia.
www.unipamplona.edu.co
Teléfono: 6075685303

***Labview un enfoque práctico para principiantes***

ISBN: 978-628-7656-04-8
Primera edición, septiembre de 2023
Colección Tecnologías e Ingenierías


Rector: Ivaldo Torres Chávez Ph.D
Vicerrector de Investigaciones: Aldo Pardo García Ph.D

Jefe Sello Editorial Unipamplona: Caterine Mojica Acevedo
Corrección de estilo: Andrea Durán Jaimes
Diseño y diagramación: Laura Angelica Buitrago Quintero

## Prólogo

Este libro es fruto de las investigaciones y desarrollo del grupo de investigación durante el año 2021. Haciendo un recorrido de todo el material con el que contamos para clases y grupo de investigación, se plantea en conjunto con mi maestro Ivaldo Torres Chávez, la idea de escribir un libro sobre LabVIEW que fuera dirigido a estudiantes en formación profesional universitaria, sobre todo en áreas de Diseño Mecatrónico, Electrónica, Sistemas embebidos.

Pretendemos, que también sea utilizado por los profesionales en estás especialidades o egresados que se interesen en el uso del LabVIEW bajo diferentes aplicaciones. En ese momento no evidenciamos un texto en castellano, sencillo, ordenado y práctico, que acogiera la inmensa información que se encuentra en la internet sobre LabVIEW y practicas asociadas a la ingeniería. Que fuese ameno para aquellos alumnos de ingeniería que quieran dar sus inicios a la programación con LabVIEW y ver sus desarrollos desde un enfoque practico.

Por otra parte, se trataba de realizar un libro claro, práctico que fuera accesible a estudiantes con pocos conocimientos de la programación en LabVIEW y sus aplicaciones. Debíamos construir una guía práctica que consiguiera enganchar a los lectores desde el inicio, y les facilitara un camino en el desarrollo de sus propias aplicaciones con LabVIEW sin perderse en toda la información que facilita la internet, o libros demasiados técnicos y complejos se seguir.

Partiendo de que LabVIEW la herramienta que se describe en este libro, es una plataforma para el desarrollo de programas de adquisición, análisis de datos y conociendo las facilidades de esta herramienta frente a otras. Se desarrolla este libro practico compuesto de 4 unidades, las cuales van desde el inicio de programación con LabVIEW desde cero hasta aplicaciones prácticas con sistemas embebidos, sin dejar de lado LabVIEW y el internet de las cosas.

El libro que está en tus manos pretende ser una guía de aprendizaje, que te permita conocer básicamente lo que es LabVIEW a través de varias practicas sencillas resueltas y otras propuestas. Además, aprenderás a manejar algunos otros softwares

como PROTEUS y manejaras dispositivos electrónicos como leds, sensores de diferentes tipos, LCD, motores de corriente continua, motores paso a paso, servomotores y como no adentrarse en el mundo del internet de las cosas (IOT), comunicando tus proyectos en LabVIEW con el internet. Todo ello partiendo de unos conceptos básico de programación y electrónica básica.

La mayor virtud en el área de la educación es poder compartir el conocimiento y la información, espero que esta herramienta ayude a aquellas personas interesadas en el aprendizaje de este lenguaje de programación.

Ivaldo Torres Chávez

Espero que esta aventura que está a punto de comenzar, les dé un panorama, de todas las aplicaciones de ingeniería que se pueden desarrollar con LabVIEW, así como nos sucedió cuando abordamos el inicio de este libro.

Diego Armando Mejía Bugallo

Como estudiante y apasionado de la programación, espero que esta herramienta sea de gran ayuda para todos aquellos interesados en LabVIEW, un software muy práctico que ayudará a desarrollar la lógica y permitirá adquirir nuevas habilidades.

Wilson Andrés Villamizar Suárez

**Convenciones utilizadas en este libro**

A lo largo de este libro se desarrollan un conjunto de prácticas de acuerdo a las temáticas. Además, se proponen otras con el fin de mejorar tu práctica en la programación con LabVIEW. Si tiene la posibilidad de acceder al software de simulación Proteus a través de una licencia o institución, podrás simular algunas prácticas en conjunto con LabVIEW. Las prácticas sugeridas se representan de dos maneras:

Las señalizadas con este símbolo, son de nivel básico y no comprenden excesiva dificultad.

Las denotadas con este logotipo se proponen otro tipo de prácticas las cuales implican un mayor esfuerzo. Afianzándote definitivamente en el tema tratado.

Con este indicativo se profundiza en algún aspecto relatado del texto próximo.

**Agradecimientos**

Ivaldo Torres Chávez

De manera personal quiero agradecer a Dios, por darme salud y vida para poder llevar a cabo todos los proyectos que me he propuesto, a mi familia por acompañarme y apoyarme en cada fase de mi vida profesional y educativa.

A mi Universidad de Pamplona, la institución que llevo en el corazón por todo lo que ha significado para mi vida, por las alegrías que me ha dado durante todo el proceso pedagógico y laboral.

Al grupo de investigación LOGOS, el cual presentó un gran interés en el desarrollo de este proyecto y a quienes destaco por su gran interés en el ámbito investigativo.

Y por último agradecer a mi equipo de trabajo, Diego y Andrés, que demostraron un gran interés en el momento de realizar este proyecto y con quienes compartí grandes momentos durante el desarrollo del mismo.

Diego Armando Mejía

En primer lugar, agradecer a la Universidad de Pamplona, por el acompañamiento y apoyo en el desarrollo de este ejercicio.

A mi maestro Ivaldo Torres, por el apoyo guía y toda la experiencia compartida.

También debo reconocer la ayuda de mi hermano Abelardo Mejía porque, en los momentos más difíciles de este proyecto, me apoyaba en todo sentido y me brindaba otros espacios de relajo para despejar la mente e iniciar nuevamente con energías recargadas.

Finalmente, reconocer la paciencia infinita de mi esposa y mis hijos durante estos últimos meses.

Wilson Andrés Villamizar Suárez

Inicialmente, a la Universidad de Pamplona por ser mi alma máter, al ingeniero Diego Armando Mejía, profesor y amigo quién, confió en mi para este proyecto, en mis habilidades y capacidades.

A mi familia, Emeli, Carolina y Angelica, quienes son mi motor, mi apoyo incondicional y las personas que más amo, me apoyaron siempre y más aún, durante el desarrollo de mis estudios académicos de pregrado.

A mi prima Mary que me ayudo con todos los procesos dentro de la Universidad, estando pendiente de y mi desarrollo estudiantil. Y por último y no menos importante, a mi pareja Gianella, que, con su paciencia, amor y comprensión, me apoyó durante los últimos y más difíciles momentos de mis estudios académicos.

**Marcas registradas**

El nombre y el logo de Arduino son marcas registradas por el equipo de Arduino en todos los países.

El nombre y el logo de Proteus ISIS, VSM y ARES son marcas registradas por Labcenter Electronics en todos los países.

El nombre y logo de LabVIEW son marcas registradas por National Instruments en todos los países.

## GLOSARIO

Analógico: Analógica quiere decir que la información, la señal, para pasar de un valor a otro pasa por todos los valores intermedios, es continua, es cualquier sistema cuyas señales se representan con valores continuos, es decir, que admite números o valores infinitos.

Arduino: Arduino es una compañía de desarrollo de software y hardware libres, así como una comunidad internacional que diseña y manufactura placas de desarrollo de hardware para construir dispositivos digitales y dispositivos interactivos que puedan detectar y controlar objetos del mundo real

Condicionales: Las condicionales en programas son grupos de sentencias o sentencias individuales que te permiten condicionar la decisión entre la elección de una opción y otra.

Delay: Es una función que hace que el procesador espere. Por ejemplo, esta espera permite no hacer nada y esperar hasta la ejecución de la siguiente instrucción durante un retardo de tiempo definido.

Diagrama de Bloques: El diagrama de bloques es la representación del funcionamiento interno de un sistema, que se hace mediante bloques y sus relaciones, y que, además, definen la organización de todo el proceso interno, sus entradas y sus salidas. Un diagrama de bloques de procesos de producción es utilizado para indicar la manera en la que se elabora cierto producto, especificando la materia prima, la cantidad de procesos y la forma en la que se presenta el producto terminado.

Digital: La señal digital, en cambio, va “a saltos”, pasa de un valor al siguiente sin poder tomar valores intermedios. Una señal analógica es continua, y puede tomar infinitos valores.

Drivers: Un controlador de dispositivo o manejador de dispositivo es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz para utilizar el dispositivo.

Entorno de Desarrollo: Un entorno de desarrollo integrado o entorno de desarrollo interactivo, en inglés Integrated Development Environment, es una aplicación informática que proporciona servicios integrales para facilitar al desarrollador o programador el desarrollo de software.

Indicador: Un indicador es una característica específica, observable y medible que puede ser usada para mostrar los cambios y progresos que está haciendo un programa hacia el logro de un resultado específico

Industria Tecnológica: La tecnología industrial es el uso de la ingeniería y la manufactura para hacer una producción más rápida, simple y más eficiente. El campo de la tecnología industrial emplea a personas creativas y técnicamente competentes que pueden ayudar a una empresa lograr una productividad eficiente y rentable.

Instalación: La instalación de los programas computacionales es el proceso fundamental por el cual los nuevos programas son transferidos a un computador con el fin de ser configurados, y preparados para ser desarrollados. Un programa recorre diferentes fases de desarrollo durante su vida útil.

IoT: El Internet de las cosas describe objetos físicos con sensores, capacidad de procesamiento, software y otras tecnologías que se conectan e intercambian datos con otros dispositivos y sistemas a través de internet u otras redes de comunicación

ISO: Un archivo ISO, también conocido como imagen ISO, es un tipo de archivo que se utiliza para almacenar una copia exacta de un sistema de ficheros de una unidad óptica. Esto quiere decir que, si copias un CD, DVD o Bluray utilizando este formato, la copia resultante será un clon exacto de esa unidad óptica, y cuando lo montes en el ordenador será como si estuvieses utilizando el disco original.

Iteraciones: es una estructura que se utiliza para ejecutar un bloque de código LabVIEW repetidamente hasta que se cumple una condición determinada.

LabVIEW: LabVIEW es una plataforma y entorno de desarrollo para diseñar sistemas, con un lenguaje de programación visual gráfico pensado para sistemas hardware y software de pruebas, control y diseño, simulado o real y embebido.

Lenguaje de Programación: En informática, se conoce como lenguaje de programación a un programa destinado a la construcción de otros programas informáticos. Su nombre se debe a que comprende un lenguaje formal que está diseñado para organizar algoritmos y procesos lógicos que serán luego llevados a cabo por un ordenador o sistema informático, permitiendo controlar así su comportamiento físico, lógico y su comunicación con el usuario humano.

Motor paso a paso: es un dispositivo electromecánico que convierte una serie de impulsos eléctricos en desplazamientos angulares discretos, lo que significa que es capaz de girar una cantidad de grados (paso o medio paso) dependiendo de sus entradas de control.

Protocolo de Comunicación: En informática y telecomunicación, un protocolo de comunicaciones es un sistema de reglas que permiten que dos o más entidades de un sistema de comunicación se comuniquen entre ellas para transmitir información por medio de cualquier tipo de variación de una magnitud física.

Puerto Serial: Un puerto serie o puerto en serie es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadoras y periféricos, donde la información es transmitida de forma secuencial.

Sistema Embebido: Un sistema embebido es un sistema de computación basado en un microprocesador o un microcontrolador diseñado para realizar una o algunas pocas funciones dedicadas, frecuentemente en un sistema de computación en tiempo real.

Sistema Operativo: Un sistema operativo es el conjunto de programas de un sistema informático que gestiona los recursos de hardware y provee servicios a los programas de aplicación de software. Estos programas se ejecutan en modo privilegiado respecto de los restantes.

Términos y Condiciones: Son las cláusulas legales que establecen la forma en la que podés usar la información y acceder a los contenidos de una página web o de una aplicación.

Contenido

## Lista de Figuras

# CAPÍTULO 1

## 1.1 Inicio Con LabVIEW Desde Cero

En este primer capítulo vamos acercarnos con decisión, al software de LabVIEW y a la instalación de su entorno de desarrollo. Demostraremos como se lleva a cabo la instalación del software principal, pero también se indicará la forma de como instalar herramientas nuevas y drivers, todo en función de lo que requiera la aplicación a desarrolla. Seguidamente se describirá como está conformada la interfaz de programación del LabVIEW panel frontal y diagrama de bloque, con las de herramientas que posee cada panel para facilitar el desarrollo de cualquier aplicación.

Todo esto, con el fin de evidenciar aquellas herramientas o características que nos ayudarán a comprender mejor el entorno de trabajo de LabVIEW, dichas herramientas serán de gran ayuda para desenvolvernos durante todo el desarrollo de esta guía, partiendo de esto es necesario comprender inicialmente que es el software y para qué sirve.

## 1.2 ¿Qué Es LabVIEW Y Para Qué Sirve?

LabVIEW (Laboratory Virtual Instrumentation Engineering Workbench) es un entorno de desarrollo y diseño de sistemas con un lenguaje visual gráfico. LabVIEW utiliza el lenguaje G (lenguaje gráfico) que acelera la productividad o desarrollo de programas para una mejor eficiencia en el desarrollo de sistemas.

Es un software creado por la empresa National Instruments en 1976 y sacado al mercado en 1986.

Al desarrollar un programa en LabVIEW, se crea un archivo de extensión VI, que contiene una interfaz de código (diagramas de bloques) y de una parte gráfica.

Una vez finalizada la programación, el usuario podrá usar la parte gráfica para controlar el sistema, junto con los diagramas de bloques que son el código de programación.

Actualmente, el software de programación LabVIEW se puede utilizar en los sistemas operativos Microsoft Windows, Mac OS X, GNU/Linux.

LabVIEW es un software muy intuitivo, que tiene las herramientas para aprender de el de una forma sencilla, la programación al ser en diagramas de bloques, es de mayor facilidad a la hora de aprender para los programadores o estudiantes.

Evidentemente, tiene su grado de dificultar si desarrollamos programas orientados a la automatización, la adquisición de datos y manejo de los mismos, el control y demás conocimientos, si se requiere un mayor conocimiento en las diferentes áreas específicas.

LabVIEW es principalmente utilizado por los ingenieros para el manejo de datos, la comunicación entre una computadora y un aparato o circuito externo es imprescindible para las aplicaciones que se le pueden dar al software, por lo que LabVIEW puede comunicarse con interfaces como:

- Puerto serial
- Puerto paralelo
- GPIB
- PXI
- VXI
- TCP/IP
- Irda
- Bluetooth
- USB
- OPC

Entre otros.

LabVIEW se presenta como una herramienta adecuada para proyectos grandes o pedagógicos, siendo un software que incorpora conocimientos de mecánica, robótica, programación, electrónica, entre otros.

La NASA, Boeing, Ford, Intel y Siemens, son empresas y agencias que suelen utilizar este software para sus distintas actividades.

Programar es sumamente importante cuando nos referimos a proyectos en el área de la automatización o el control industrial, es un área de especialización para los ingenieros mecatrónicos, el saber programar es muy importante en el mundo moderno debido a múltiples factores como, por ejemplo:

- Demanda laboral: La programación es una de las habilidades más demandadas en el mercado laboral actual. Cada vez son más las empresas que necesitan programadores para desarrollar software y aplicaciones.

- Automatización: La programación es esencial en la automatización de procesos, lo que ayuda a aumentar la eficiencia y reducir errores en diversos campos como la industria, la logística, el comercio y la atención al cliente.

- Innovación: La programación es una herramienta fundamental para la innovación. La capacidad de crear nuevos productos y servicios tecnológicos se basa en la capacidad de programar y desarrollar software.

- Emprendimiento: La programación es esencial para el emprendimiento tecnológico, ya que permite a los emprendedores crear y desarrollar sus propias aplicaciones y productos tecnológicos.

- Competencias digitales: En el mundo moderno, la mayoría de los trabajos requieren habilidades digitales. La programación es una de las habilidades más importantes en este ámbito.

“La potencia está en el software” Una frase muy célebre de LabVIEW, que hace referencia a la capacidad e importancia que puede tener un programa en un proyecto.

## 1.3 Configuración E Instalación

Para la instalación y configuración del LabVIEW existen varias formas, entre estas se tiene instalación por memoria USB, instalación por CD o instalación en línea desde la página de la Nationals Instrument.

### 1.3.1 **Instalación de LabVIEW**

Ingresar a la ruta donde se encuentre el instalador y ejecutar el autorun (install.exe), si el archivo aparece con la extensión .ISO, dependerá de la versión de Windows si requiere un montador de imágenes ISO o no, Windows 10 y Windows 11 trae esta herramienta incorporada.

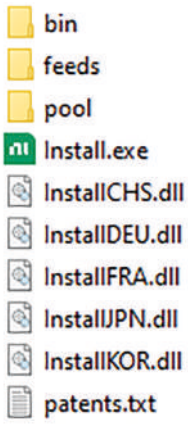



*Figura 1. Archivos de instalación del LabVIEW. Autoría Propia*

A continuación, en la Figura 2, debemos aceptar los términos de la instalación, estos términos son un protocolo que hay que aceptar debido a las políticas internas de la empresa, al aceptar la licencia la cual recomendamos leer, podremos continuar la instalación del software.

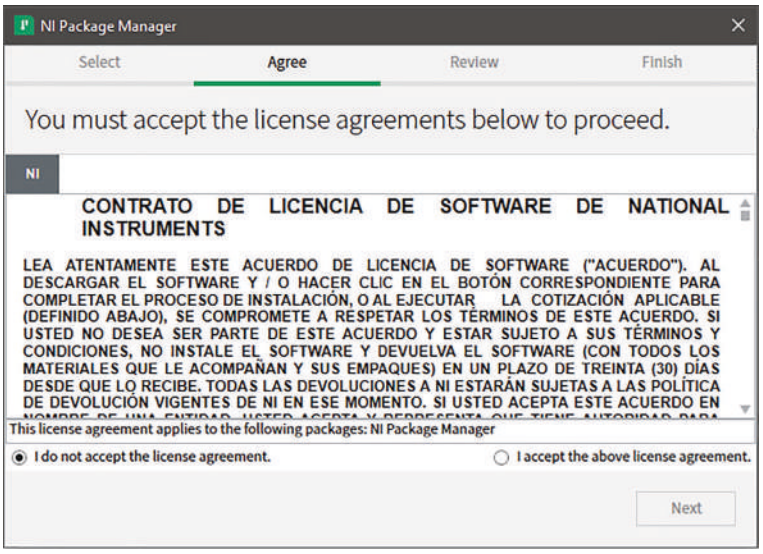



Figura 2. Términos de instalación LabVIEW. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Una vez aceptados los términos, nos aparecerá una ventana similar a la figura 3, donde se evidencias los productos que nos ofrece la instalación, estos productos pueden ser comprendidos como servicios, herramientas, protocolos, drivers y demás características que nos brinda el software.

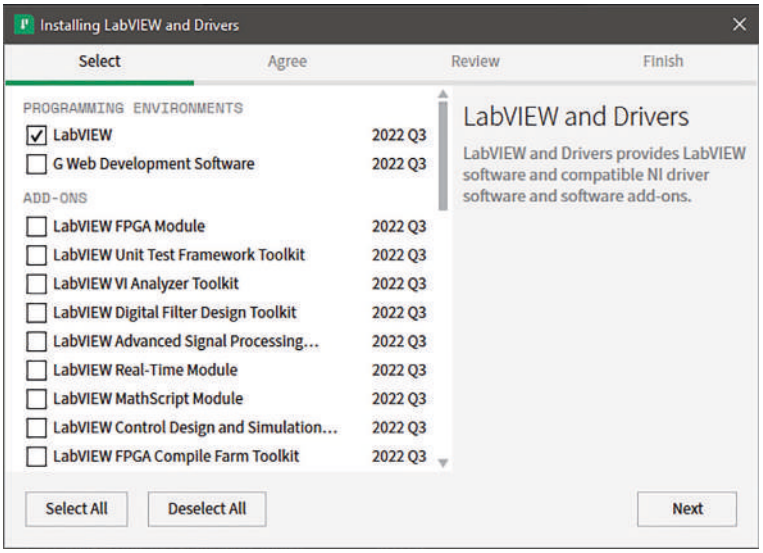



Figura 3. Selección de módulos a instalar. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Desplegar en el signo + y escoger los productos que se desean instalar y dar clic en Next, se debe tener en cuenta que, entre más herramientas deseemos instalar, mayor será el consumo del espacio dentro de nuestro computador, por lo que la instalación del software podrá consumir entre 30 GB y 120 GB de memoria, debido a esto se recomienda revisar el espacio disponible en el computador que se está instalando el software.

Los complementos son de gran importancia para el desarrollo de las prácticas y si tenemos un énfasis en la investigación y en la competitividad, será recomendable instalar todos los complementos e investigar sobre su aplicación, uso, características y demás información que puedan ayudar a tener un conocimiento altamente competitivo en esta área de investigación.

Es necesario recalcar que, la velocidad de procesamiento y rendimiento del software también estará dada por el tipo de disco que comprende su computador, un disco SSD o MVME tendrá un rendimiento mayor que un disco de tipo HDD, sin embargo, LabVIEW está tan bien optimizado que en un disco HDD funcionará bien, solo que tendrá un rendimiento un poco pequeño en comparación con un SSD o MVME.

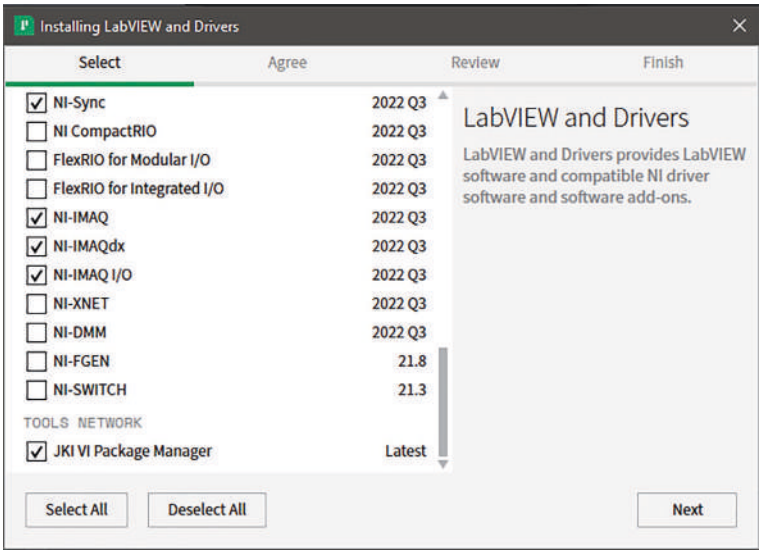



Figura 4. Selección de driver a instalar. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Nos aparece la siguiente ventana, aceptamos y damos clic en Next.


Figura 5. Términos de licencia. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Aceptar los términos de licenciamiento.

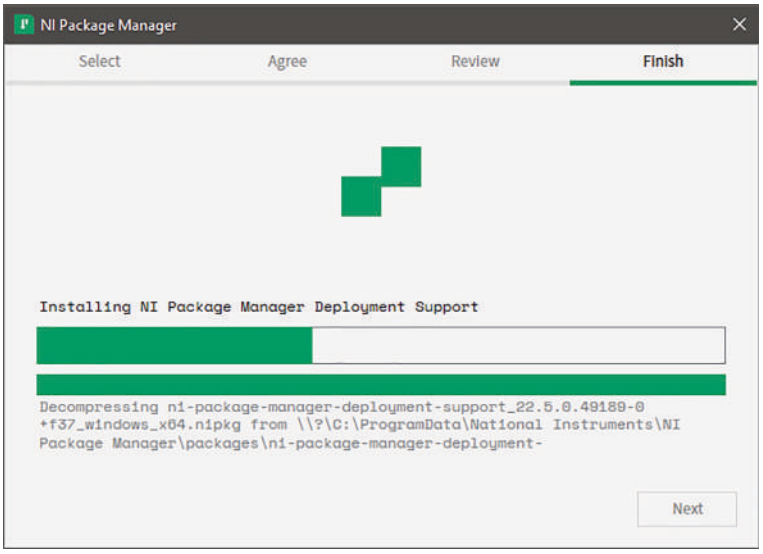



Figura 6. Progreso de instalación. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Es posible que por cada paquete que este instalando tengamos que dar clic en Next cuando termine cada instalación.

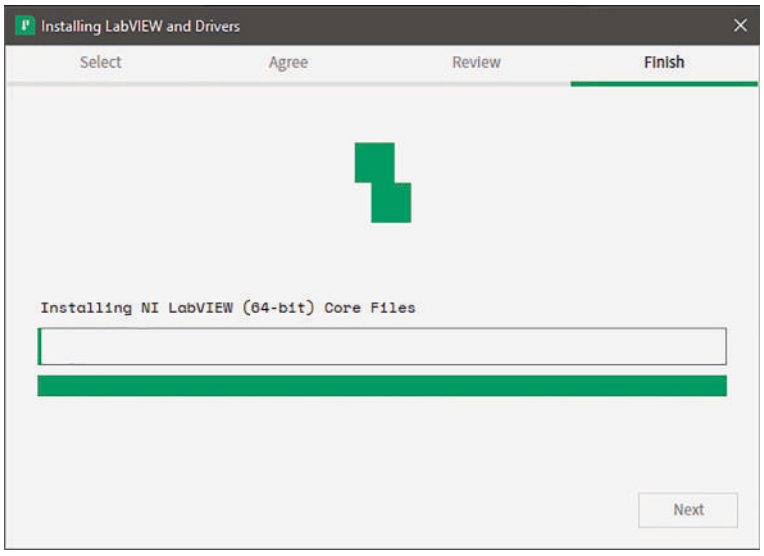



Figura 7. Etapa final del progreso de instalación. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la siguiente ventana nos informa cuales paquetes de los seleccionados se instalaron satisfactoriamente. Dar Clic en finalizar para terminar la instalación.

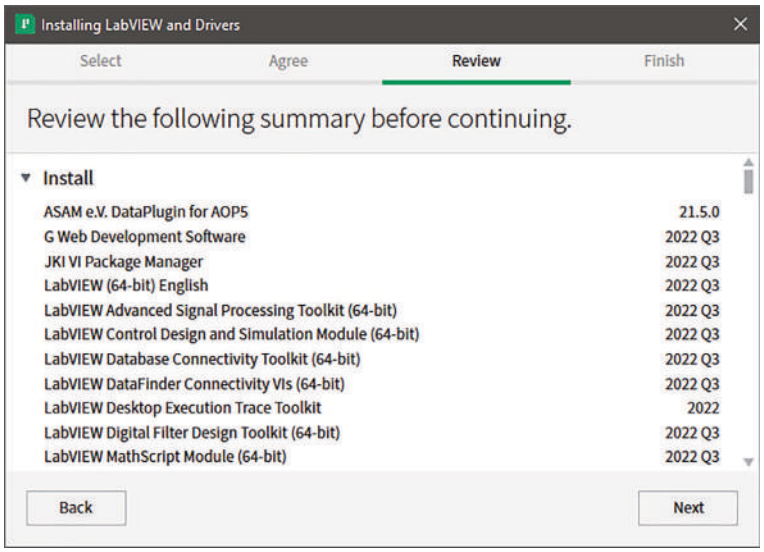



Figura 8. Módulos instalados. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

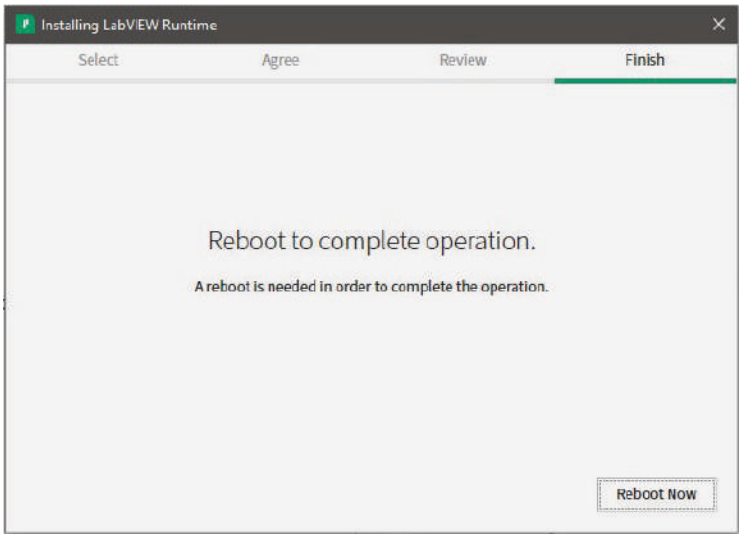



Figura 9. Finalización de la instalación. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Reiniciamos el equipo para finalizar la instalación.

### 1.3.2 **Instalación de Toolbox de LabVIEW**

Para la instalación de Toolbox de LabVIEW, se utiliza la herramienta NI Package Manager, la cual puede gestionar los paquetes deseados como se observa en la imagen.

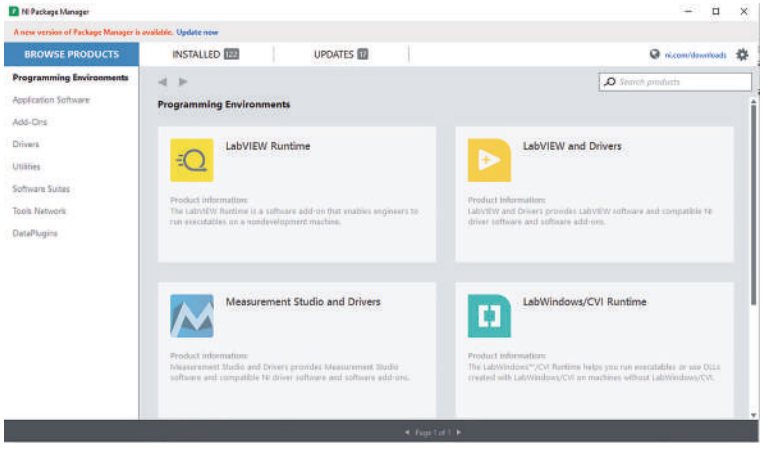



Figura 10. Interfaz de NI Package Manager. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Luego se siguen las indicaciones del instalador, de acuerdo a lo que va indicando el asistente.

Dentro de las opciones se pueden instalar entornos de programación, aplicaciones, complementos, entre otras opciones.

### 1.3.3 **Instalación De Driver Para Comunicación Con LabVIEW**

La instalación de drivers en LabVIEW, depende de la aplicación que lo requiera, para esto también se utiliza la herramienta NI Package Manager, la cual puede gestionar los drivers deseados, las comunicaciones tienen sus respectivos drivers, por lo que podemos recomendar que, dependiendo de la necesidad o el tipo de comunicación, se realice la instalación de su respectivo driver.

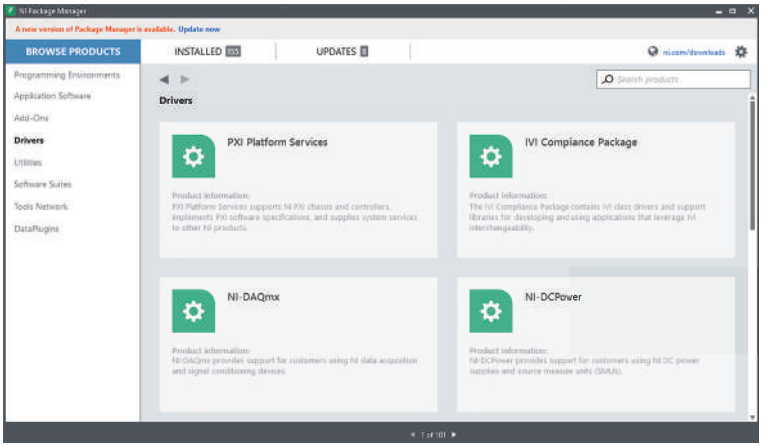



Figura 11. Interfaz NI Package Manager – Driver. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Luego se siguen las indicaciones del instalador, de acuerdo a lo que va indicando el asistente.

## 1.4 Descripción De La Interfaz De LabVIEW

Al ejecutar el software LabVIEW, nos aparece una ventana como la siguiente:

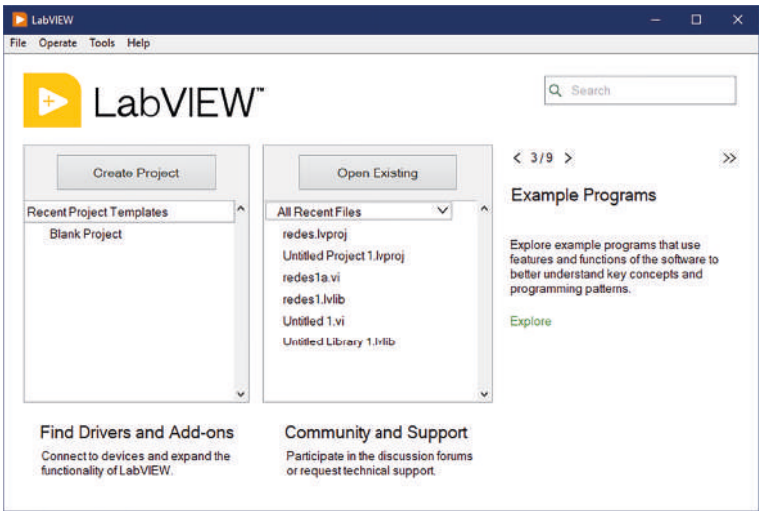



Figura 12. Pantalla de bienvenida de LabVIEW. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Aquí podemos elegir del menú la opción de crear un nuevo proyecto, abrir un proyecto existente, crear un VI en blanco o un VI desde una plantilla. Si seleccionamos crear un nuevo proyecto nos aparece una nueva interfaz como se puede ver en la imagen.

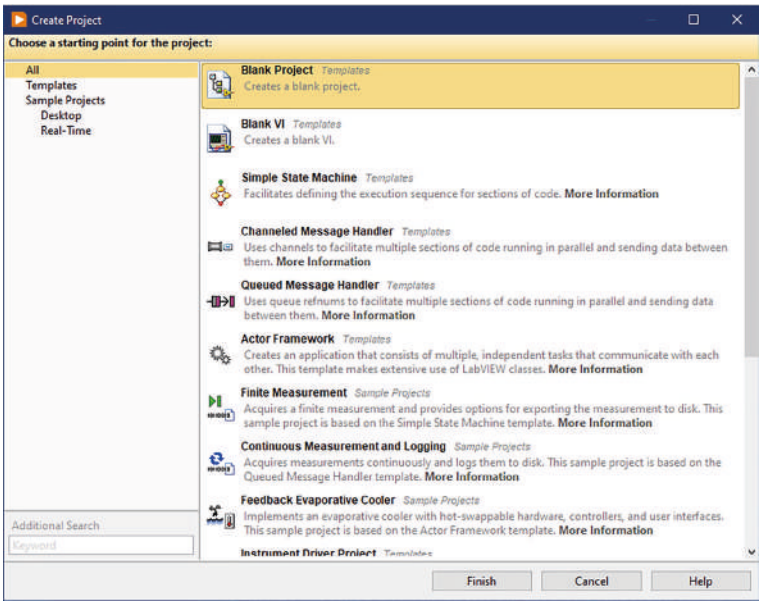



Figura 13. Pantalla de crear proyecto de LabVIEW. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Tenemos muchas opciones para crear un proyecto, la más adecuada es Blank Project o Proyecto en blanco, donde nos aparecerán dos ventanas, el panel frontal y el diagrama de bloques.

### 1.4.1 **Panel Frontal**

En el panel frontal, es donde vamos a manejar la interfaz que será el intermediario entre el usuario, el programador y la máquina.

El panel Frontal está constituido por una serie de objetos gráficos, que son parte del software y que nos presentan objetos que varían gráficamente en razón a la programación que se realice, aquí encontraremos indicadores y controles de ayudan a una visualización rápida, los componentes vienen ya por defecto, sin embargo, el usuario podrá crear sus propios indicadores dentro de unas opciones mucho más avanzadas, la siguiente ilustración nos muestra el panel frontal:

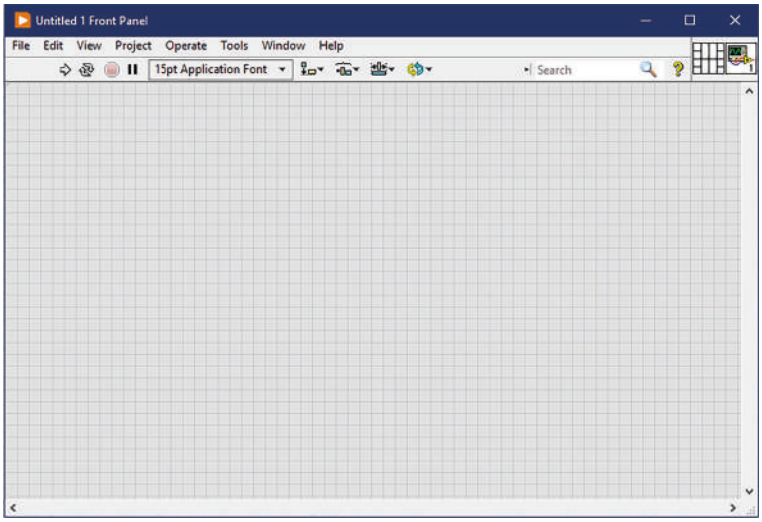



Figura 14. Panel frontal del programa en LabVIEW. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la parte superior de la ventana encontraremos la barra de herramientas, donde encontramos botones, indicadores y otro tipo de características que nos ayudan a controlar la interfaz.


Figura 15. Barra de herramientas del panel frontal. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

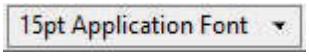
La barra de herramientas estará disponible dependiendo en la sección donde estemos trabajando, si es el panel frontal o el diagrama de bloques.

 Este botón ejecutará el archivo una única vez.

 Ejecutará nuestro programa de manera infinita, mediante un bucle.

 El botón abortar sirve para salir inmediatamente de la simulación.

 El botón de Pausa/Continuar sirve para realizar una pausa, para continuar el botón debe presionarse nuevamente



El Anillo de Fuentes. Sirve únicamente para seleccionar la fuente. El color, el tamaño y demás características de tipografia.

 El Anillo de Alineación. Es una herramienta que ayudará en la alineación de tipo vertical, horizontal o centrada.

 El Anillo de Distribución. Use esta herramienta distribución para seleccionar opciones de distribución incluyendo espacios, compresión etc. Para dos o más objetos.

 El Anillo de Dimensionamiento. Use esta herramienta para dimensionar objetos del panel Frontal.

 El Anillo de Ordenamiento. Use esta herramienta de jerarquía que ayuda a variar el orden de empalamiento.

Una de las herramientas mas importantes se encuentra en la esquina superior derecha, cuando tenemos dudas sobre como funciona el bloque, la barra de búsqueda nos indicará las especificaciones del bloque, su funcionamiento y el tratamiento de datos.

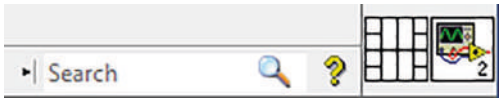



Figura 16. Barra de búsqueda y propiedades del VI. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Desde el otro icono, podremos editar no solo el tamaño de la ventana, sino una opción muy importante para evitar cualquier tipo de copia o plagio de nuestro código, el de crear una contraseña para proteger nuestro código, también podremos ver el uso de la CPU, la memoria que está usando, disco duro, rendimiento y demás

### 1.4.2 **Diagrama de bloques**

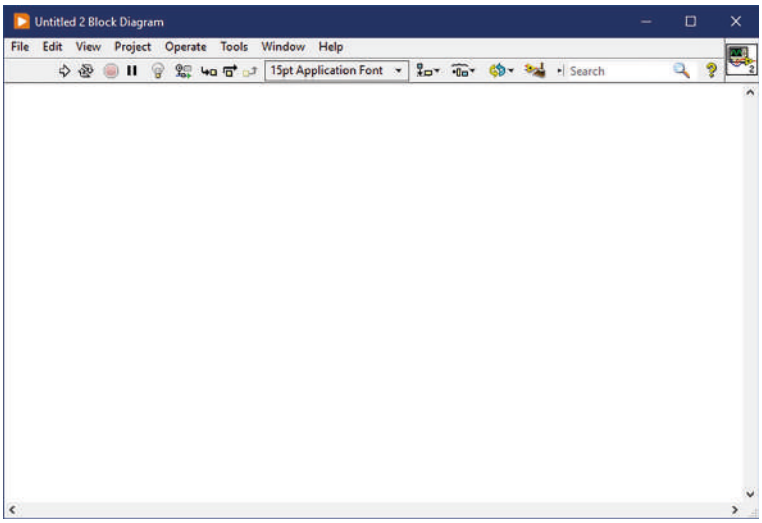



Figura 17. Diagrama de bloques en LabVIEW. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Dentro del diagrama de bloques, tendremos la barra de herramientas, esta contiene prácticamente la misma barra de herramientas que hay en el panel frontal, solo que cuenta con cuatro herramientas adicionales, las cuales son:

 El botón de animación de la ejecución. Al presionar este botón, se simulará una animación que irá recorriendo el código paso a paso, es de gran ayuda para lograr encontrar errores dentro de nuestro código.

 Modo animado, se habilita esta acción cuando la animación está ejecutándose, permitiendo ver el flujo de los datos mediante el diagrama de bloques.

 El botón de Pasar Sobre. Este botón habilita el modo paso a paso, permitiendo interactuar en cada nodo del proyecto.

 El botón de Entrar A. Con este botón podremos entrar a un ciclo, permitiendo ver los datos que están siendo tratados y ejecutados en este momento.

 El botón de Salir De. Este botón sirve para salir del ciclo, yendo hacia el siguiente nodo.

 Este botón, es de gran ayuda para códigos que se encuentran desordenados, ordenará el código de una forma legible y que ocupe poco espacio en la interfaz

## 1.5 Herramientas Útiles De La Interfaz De LabVIEW

*Tools Palette*

Esta es la paleta es muy importante a la hora de programar, una vez la activemos la tendremos disponible tanto en el panel frontal como en el de diagrama de bloques, esta herramienta ayuda a crear, modificar y depurar los archivos de extensión VI, si no está visible podemos activarla en el MENU, seleccionamos VIEW y posteriormente la opción de Tools Palette:

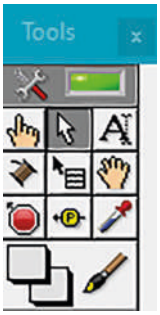



Figura 18. Tools palette. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Una vez habilitado, podemos seleccionar las herramientas de la paleta, es importante destacar que nuestro cursor, tomará la imagen de la herramienta que seleccionemos, con eso sabremos en todo momento que herramienta estamos utilizando, las herramientas son:

 Herramienta de Selección Automática, es la que viene por defecto, actuará en relación al objeto que se encuentre bajo la herramienta, ya sea una operación, posicionamiento, etiquetado o cableado.

 Herramienta de Operación. Ayuda a manipular los indicadores y controles del panel frontal

 Herramienta de Posicionamiento. Ayuda mover objetos, seleccionar o redimensionar dentro del panel frontal y los diagramas de bloques.

 Herramienta de Etiquetado. Esta herramienta ayuda a editar los textos de los objetos, aplica para ambas ventanas, la de diagrama de bloques y el panel frontal.

 Herramienta de Cableado. Esta herramienta ayuda a unir objetos por medio de cables, los tipos de cables varían en funcion del tipo de datos que trasmite.

 Herramienta de Pop-Up. Utilice esta herramienta para tener acceso al menú pop-up de un objeto al oprimir el botón izquierdo del Mouse.

 Herramienta de Deslizamiento. Ayuda a desplazarnos a través de las ventanas sin utilizar las barras de movimiento, es una herramienta muy útil cuando detenemos un proyecto demasiado grande.

 Herramienta de Puntos de Detención. Esta herramienta sirve para colocar los puntos de ruptura

 Herramienta de Pruebas. Ayuda a colocar probadores dentro de los cables en la interfaz de diagrama de bloques.

 Herramienta de Copiado de Color. Sirve para colocar colores dentro de la herramienta, nos ayuda a diseñar mejor la interfaz gráfica.

 Herramienta de Color. Ayuda a colorear un objeto, dentro de una interfaz gráfica es útil para distinguir diferentes objetos de control.

*Controls Palette*

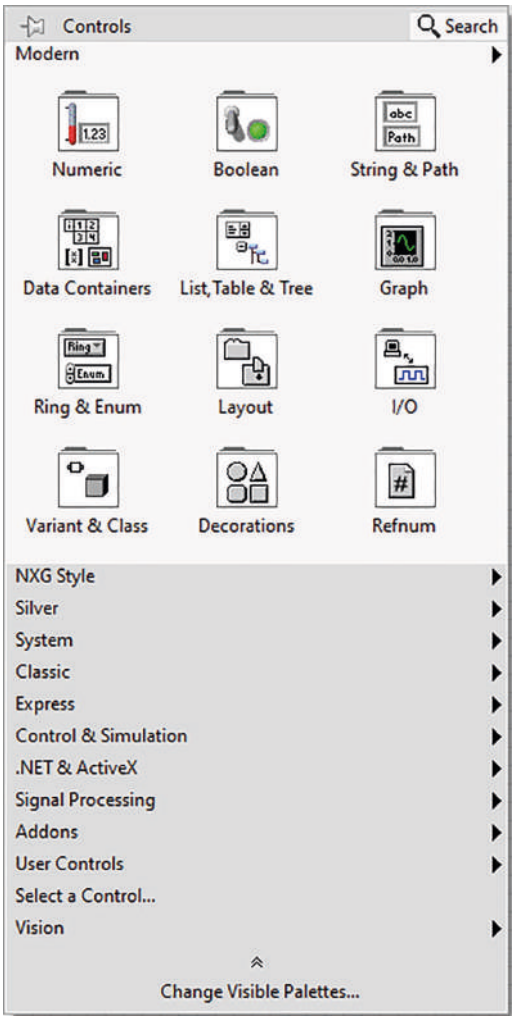
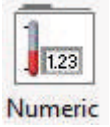



Figura 19. Controls palette. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ahora, la paleta de controles estará disponible solo en la interfaz del panel frontal, consiste en una serie de categorías de alto nivel y a su vez, cada categoría cuenta con subcategorías que dan acceso a una amplitud de objetos disponibles, para acceder a una subcategoría solo debemos colocar el mouse sobre la categoría y ya. Si por alguna razón, no se encuentra visible, simplemente en el MENU > VIEW y activamos el Controls Palette, las categorías que encontramos son:




NUMERIC (Numérico). Contiene los indicadores y controles de tipo

numérico, podemos ingresar o evidenciar, números ya sean medidos o calculados por el código de diagrama de bloques


BOOLEAN (Booleano). Contiene controles e indicadores de datos tipo booleanos, ON – OFF, 1 – 0, HIGH – LOW.


STRING (Cadenas de Caracteres). Contiene indicadores y controles de tipo cadena de texto y herramientas de tipo path, que funciona para evidenciar rutas en el disco duro, para tener en cuenta con archivos almacenados en nuestro computador

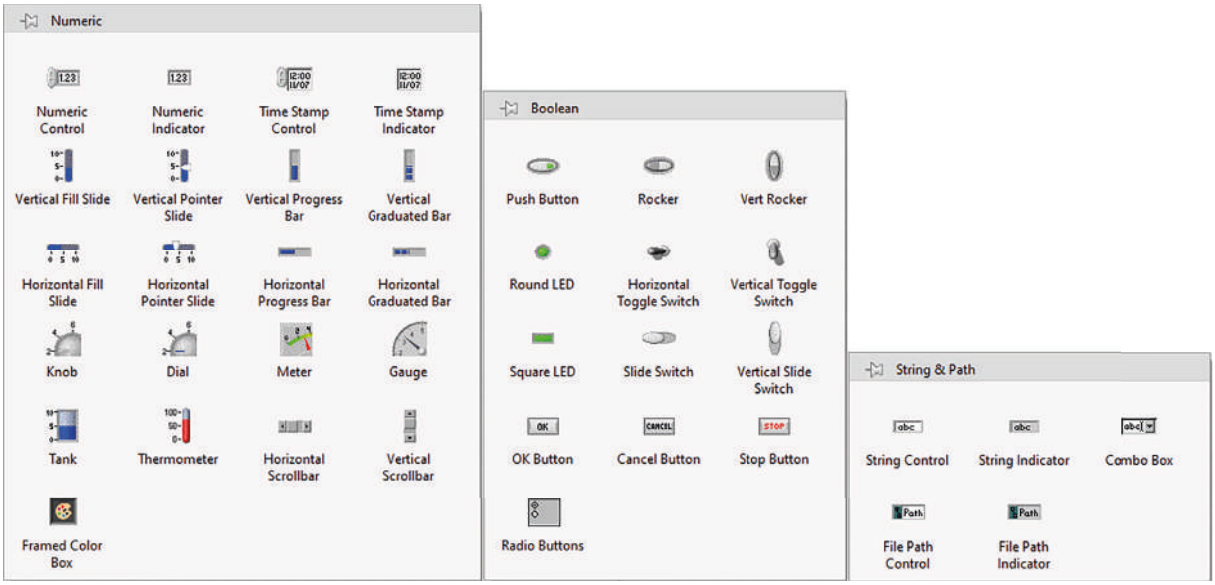



Figura 20. Categorías de la entrada Modern. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*


ARRAY, CLUSTER & MATRIX (Arreglos y Agrupamiento). Esta

categoría contiene características para trabajar con tipos de datos agrupados, ya sean arreglos, matrices o clústers.


LIST, TABLE & TREE (Listas, Tablas Y Árbol). Contiene listas, tablas, indicadores como formato de texto.


GRAPH (Gráficas). Contiene herramientas gráficas que funcionan como indicadores, es de utilidad para graficar señales o conjuntos de datos.

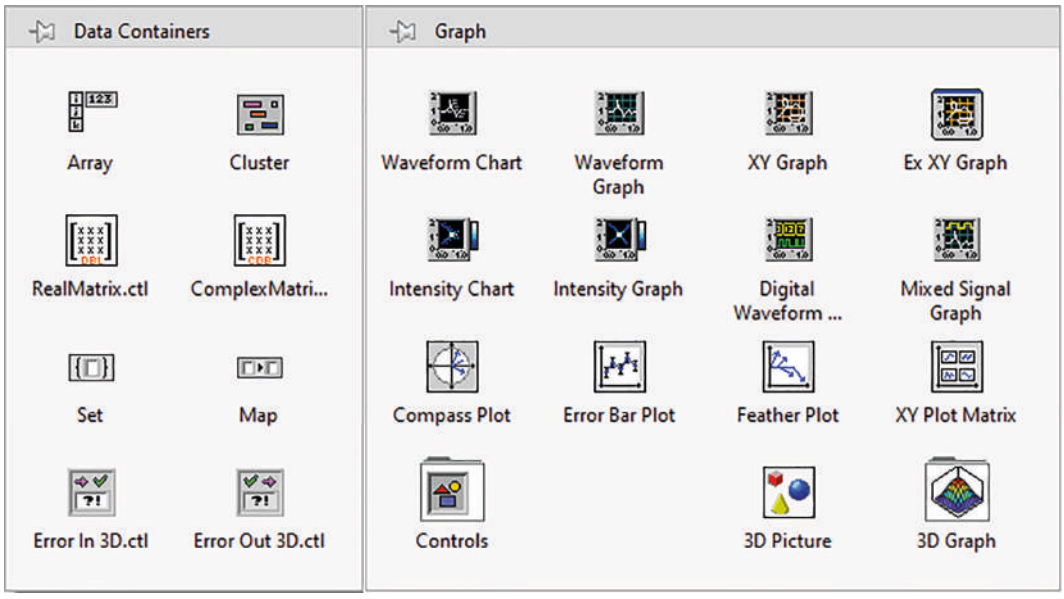



Figura 21 Categorías de la Entrada Data Containers y Graph. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

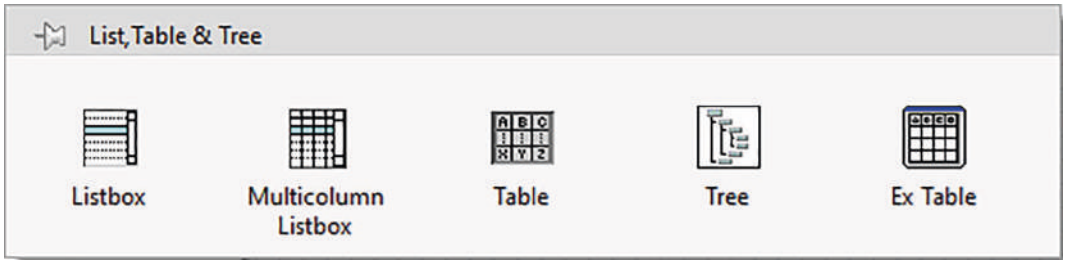
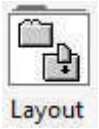



Figura 22. Categorías de la entrada Modern (LIST, TABLE & TREE). Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*


RING & ENUM (Anillo y enumerador). En esta categoría encontraremos indicadores y controles que funcionan como un menú desplegable, útil para seleccionar casos.




CONTAINERS (Contenedores). Categoría que trae herramientas de tipo contenedor o tableros, que ayudan a manejar los objetos análogos.

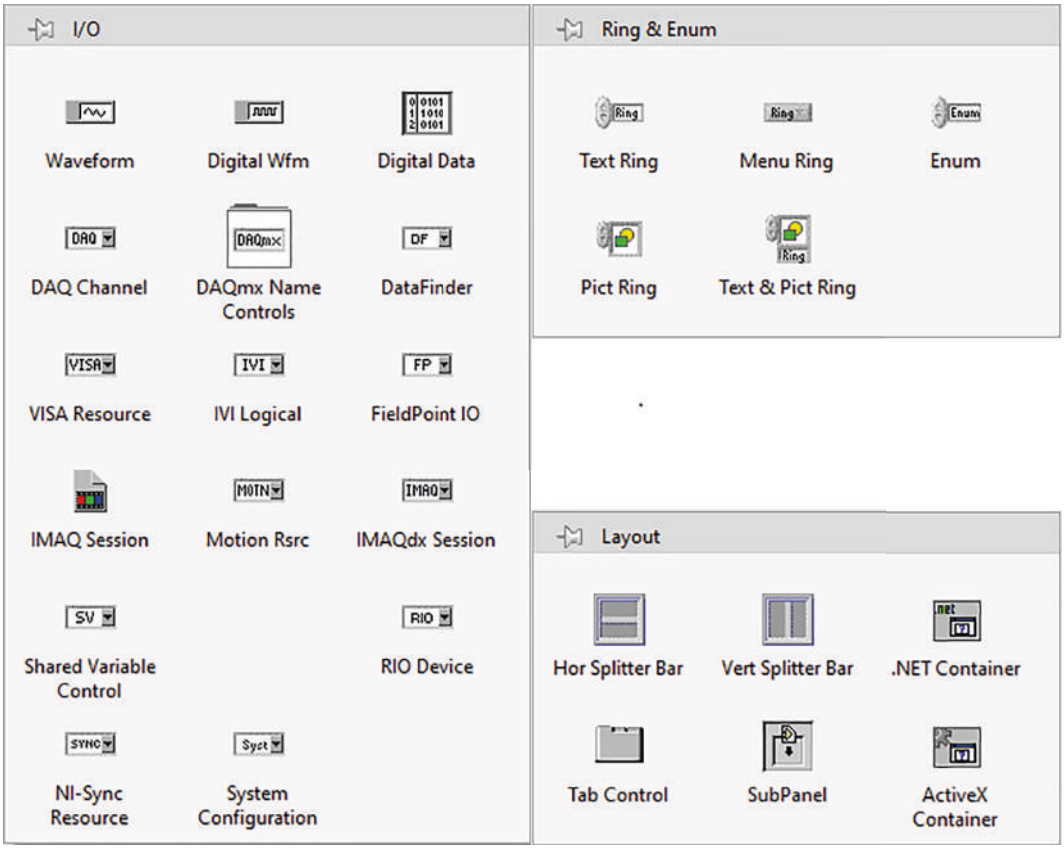
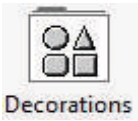



Figura 23. Categorías de la entrada Modern (RING & ENUM, CONTAINERS, I/O). Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

que

e los

instrumentos de adquisición de datos.




DECORATIONS (DECORACIONES). Es una categoría que como su nombre lo indica, ayuda a decorar la interfaz grafica del panel frontal, estos objetos no inciden en la programación, pero si en la estética.


Figura 24. Categorías de la entrada Modern (DECORATIONS). Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

***Function palette***

Los bloques se construyen utilizando la paleta de funciones, cada opción tiene sus propias subcategorías, si por alguna razón no está visible, al igual que las otras ventanas, solo debemos ingresar al MENU > VIEW y habilitar la Function palette, también podremos acceder dando clic derecho en un área libre dentro del diagrama de bloques.

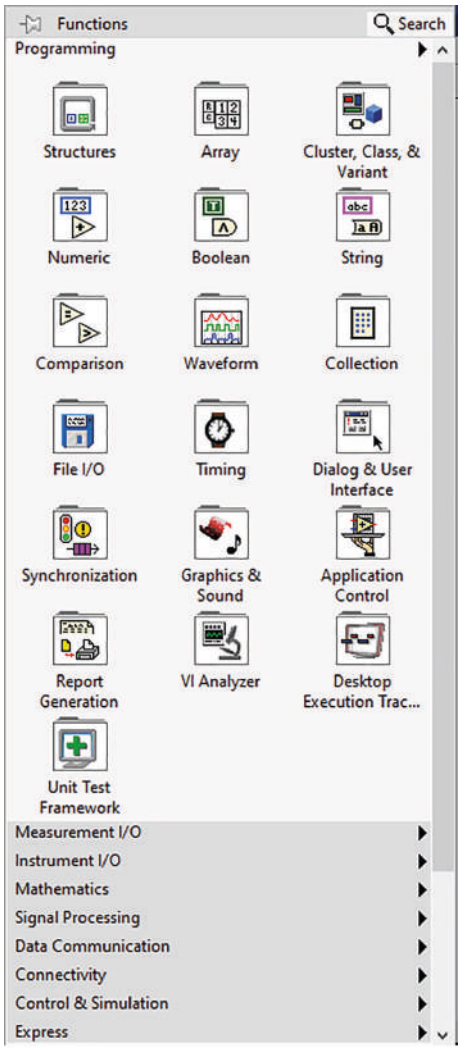
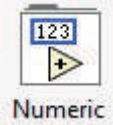


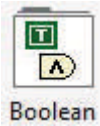

Figura 25. Paleta de funciones. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

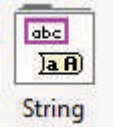
Al desplegar la categoría de PROGRAMING, podremos visualizar las categorías con las que iniciaremos dentro de la programación de bloques de nuestro proyecto, las subcategorías son:




Numeric (Numérico). En esta categoría se encuentran funciones básicas matemáticas, constantes, números complejos y todo tipo de datos numéricos.




Boolean (Booleano). Contiene las principales funciones para el manejo de datos de tipo booleano y su lógica, también encontraremos herramientas que ayudan a convertir números.




String (Cadena de Caracteres). Las funciones principales para manipular y manejar tipo de texto en cadena, podremos convertir texto a otros formatos incluyendo el numérico o path.

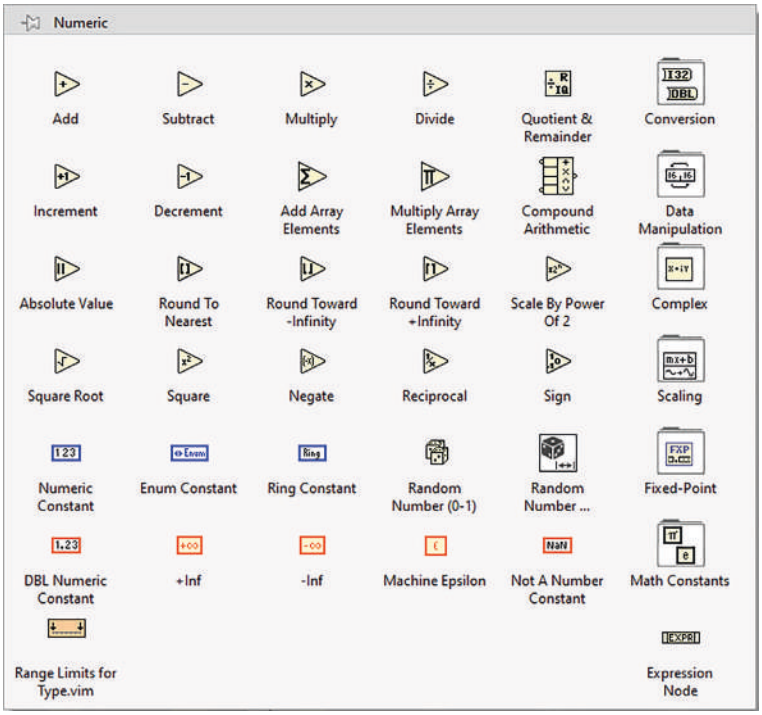



Figura 26 Categoría Numeric y sus respectivas funciones. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

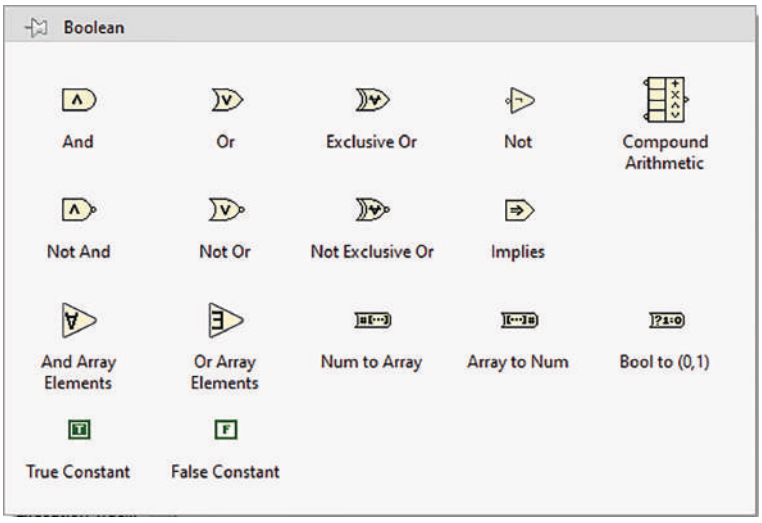



Figura 27 Categoría Boolean. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

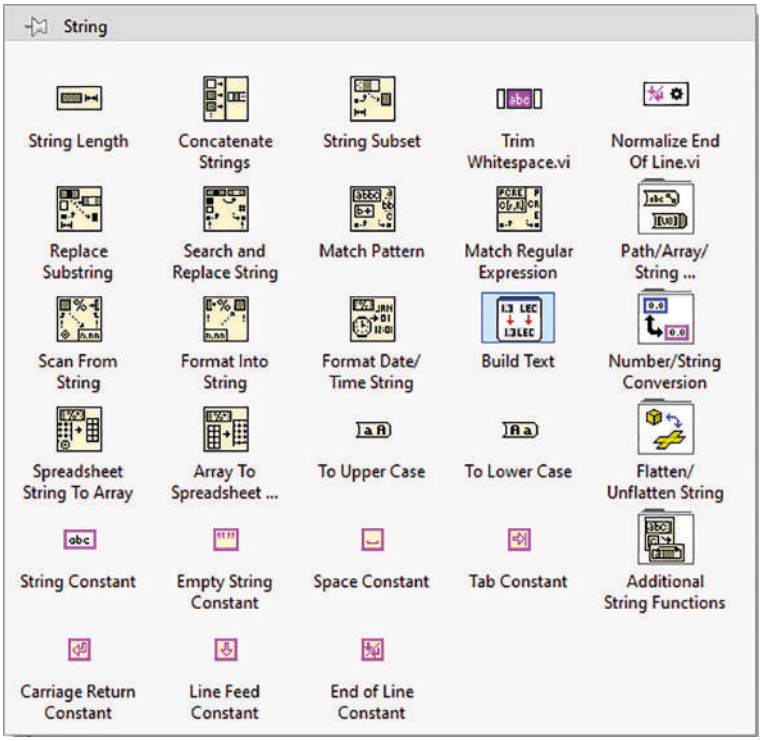



Figura 28. Categorías de la entrada String. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*


Structures (Estructuras). Esta categoría es importante, aquí encontraremos los ciclos que se suelen usar en programación, como por ejemplo el ciclo for, while, case entre otros.

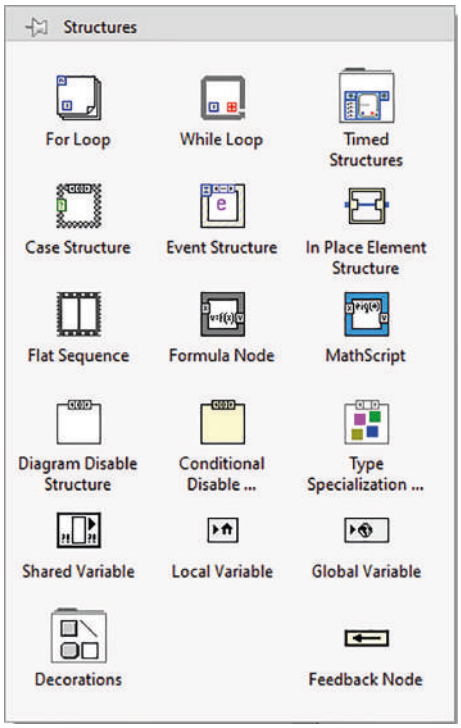



Figura 29 Categoría Structures Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*


Array (Arreglos). Contiene funciones para procesar arreglos de datos y matrices, estos arreglos constan de elementos y dimensiones, la dimensión es considerada como una longitud, una altura o una profundidad, si se habla de arreglos de dos dimensiones se contempla un alto y un ancho, si hablamos de un arreglo de tres dimensiones consideramos adicionalmente la profundidad, esto tiene múltiples aplicaciones en los campos de desarrollo de software y aplicarlo en nuestra área, será de gran ayuda para llevar a cabo la gestión y almacenamiento de datos.

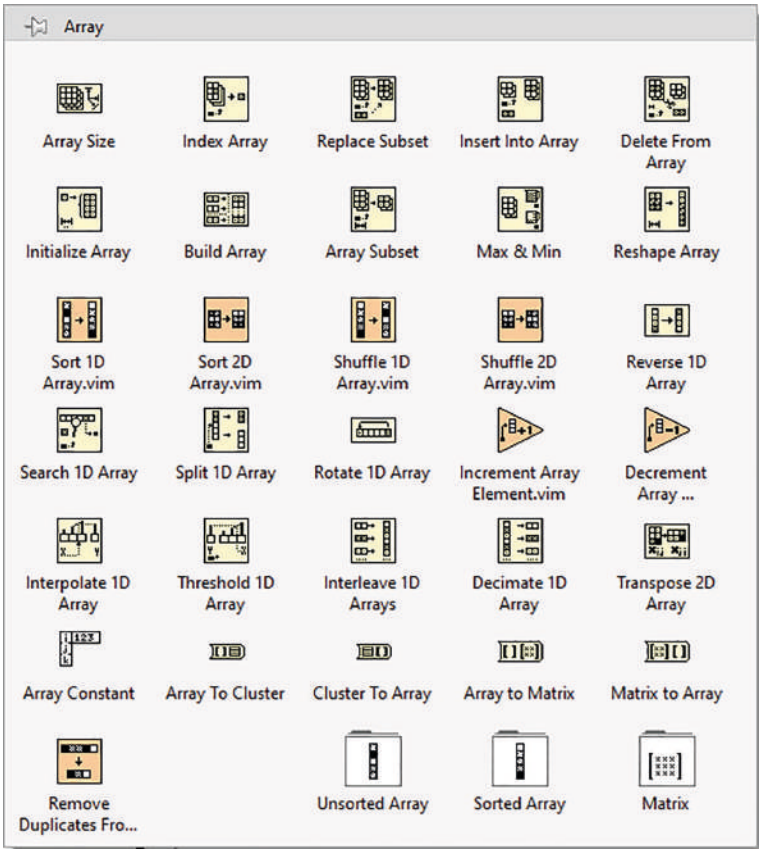
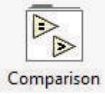



Figura 30 Categoría Arrays. Autoría Propia

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*




Comparison (Comparación). Esta categoría contiene funciones para comparar todo tipo de datos, a sean numéricos, booleanos, cadena de caracteres entre otros.

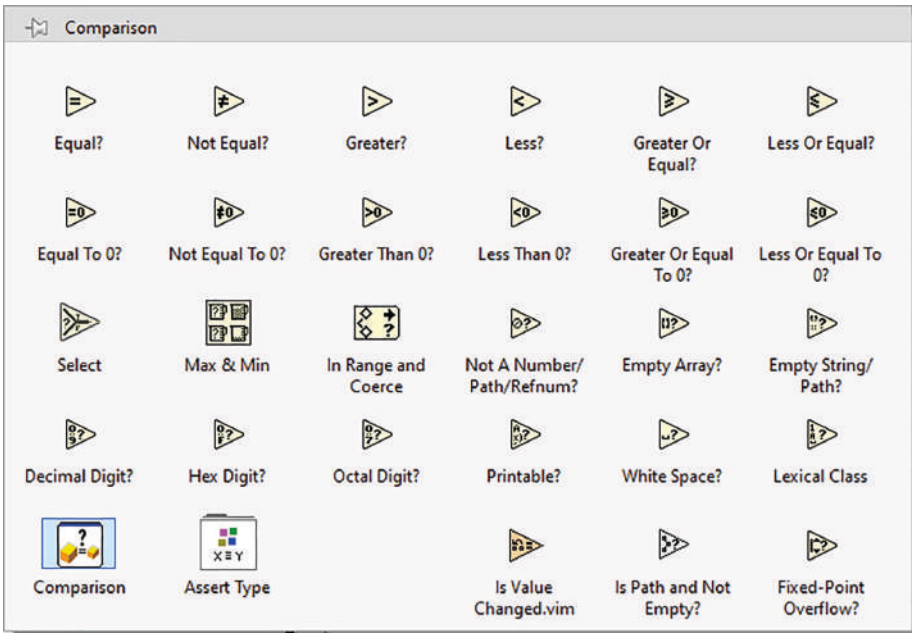
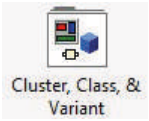
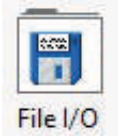



Figura 31. Categorías comparision. Autoría Propia

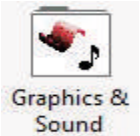
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*




Cluster & Variant (Agrupamiento y Varianza). La categoría contiene principalmente funciones para agrupar y desagrupar datos, a su vez, contiene herramientas para manejar los tiempos, como cronómetros, temporizadores, delays entre otros.

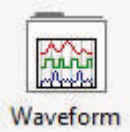



File I/O (Manejo de Archivos). Esta categoría cuenta con herramientas para guardar datos en archivos y bases de datos, es de suma importancia cuando queremos manipular datos.



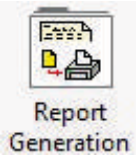

Graphics & Sound (Gráficos y Sonidos). Contiene funciones básicas para el procesamiento y manejo de sonidos e imágenes en 2D y 3D, planos polares y gráficos cartesianos.


Dialog & User Interface (Dialogo e Interface de Usuario). Esta categoría contiene herramientas que nos ayudarán a manejar cuadros de diálogos, ventanas de errores y de programación




Waveform (Formas de Onda). Contiene funciones con características para el manejo o manipulación de tipos de datos análogos y digitales.


Aplication Control (Control de Aplicaciones). Son funciones básicas de esta categoría, el control de datos para los servidores de VI, es útil para manipular los menús de LabVIEW.




Report Generation (Generación de Reportes). Ayuda a generar reportes de cualquier tipo de error, guardando la información en archivos de datos, como puede ser Excel, HTML o Word


Synchronization (Sincronización). La sincronización es importante en todo tipo de proyecto, esta categoría contiene funciones para sincronizar los lazos dentro de un proyecto.

Entradas desplegables: son clasificadas por grupos de características similares y herramientas de VI que ayudan a programar con un nivel más avanzado, dentro de esta entrada encontramos:

*Entrada Measurement I/O (Medida In/Out)*

Contiene Funciones para manejar dispositivos de adquisición y envío de datos, será de gran utilidad en las practicas que desarrollaremos próximamente con un sistema embebido especifico.

*Entrada Instrument I/O (Instrumentos In/Out)*

Contiene Funciones para administrar dispositivos o instrumentos conectados por cualquiera de los protocolos (GPIB, Serial, NI, VISA, etc).

*Entrada Vision and Motion (Movimiento y Visión)*

Manejo básicos de IMAQ y MOTION de NI (National Instruments)

*Entrada Matematics (Matematicas)*

Esta entrada contiene funciones trigonométricas, estadísticas, de algebra, calculo lineal, formulas y logaritmos.

*Entrada Signal Procesing (Procesamiento de Señal)*

Tratamiento de señales por filtro, ajuste de curvas y análisis del espectro.

*Entrada Data Comunication (Comunicación de datos)*

Aplica para comunicación TCP, DDE y serial.

*Entrada Conectivity (Conectividad)*

Si requerimos de conectar dispositivos en puertos, controlar el ingreso de la información o demás, tendremos disponibles herramientas para esta ocasión.

*Entrada Express (Expreso)*

Contiene una serie de herramientas de tipo Express, facilitando la programación de la aplicación.

*Entrada Select a VI (Seleccionar VI)*

Ayuda a importar archivos con extensión .vi, guardados en un disco duro para ingresar de forma eficiente un subprograma a lo que conocemos como programación modular, dejando funciones programadas por defecto y reutilizadas en otros programas.

# CAPÍTULO 2

## 2.1 Lenguaje De Programación De LabVIEW

En este segundo capítulo abordaremos la terea de comprender los aspectos básicos del lenguaje de programación de LabVIEW y abordaremos en detalle la lógica de programación y funciones las cuales maneja para facilitar en desarrollo de aplicaciones. Se describirán algunos ejemplos de acuerdo a la respectiva función que estemos abordando, sea Controles e indicadores, elementos booleanos, uso de Select, uso de cadenas de texto, uso de la estructura Case, uso de Array, uso de Arrays con String, ciclo For, shift Register, ciclo While, variables locales, entre otras.

Comprender los conceptos básicos es fundamental para cualquier área del conocimiento, y en particular para el aprendizaje de un software como LabVIEW. Los conceptos básicos son los cimientos sobre los cuales se construye todo el conocimiento posterior. Sin una comprensión sólida de los conceptos básicos, podrás pueden tener dificultades para comprender conceptos más avanzados y para aplicarlos en la resolución de problemas prácticos.

En este caso, los conceptos básicos incluyen en como comprender los bloques de construcción fundamentales de los programas, como los bucles, las estructuras de control, las variables y las funciones. También es importante tener una comprensión clara de cómo se representa visualmente la información en LabVIEW, mediante la creación de gráficos y la utilización de diferentes tipos de datos.

Teniendo en cuenta esto, podrás identificar errores y solucionar problemas con más eficacia. En lugar de simplemente tratar de memorizar la forma de escribir un programa o la sintaxis de un comando, con los conceptos básicos se podrán abordar los problemas de manera más sistemática, analizando las causas subyacentes de los errores y utilizando su comprensión de los conceptos para encontrar soluciones.

Acabaremos este segundo capítulo con un par de ejercicios los cuales sintetizan el uso de todas las funciones enunciadas anteriormente.

## 2.2 Introducción Al Concepto De Programación

Una de las principales ventajas del LabVIEW es el tiempo que se tarda un programador en desarrollar un programa, siendo una facilidad que ofrece a los desarrolladores no expertos. Además de manejar una gran cantidad de paquetes que permiten combinar este software con todo tipo de hardware, como tarjetas de adquisición de datos, controladores, autómatas programables, sistemas de embebidos.

Ofreciendo una característica para programar mediante bloque, ahorrando líneas de código y, por ende, complicación en temas de código y tiempo de escritura, este software es una gran herramienta para el aprendizaje de programación en el área de la ingeniería mecatrónica.

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) fue creado por la empresa National Instruments para funcionar sobre máquinas MAC, salió al mercado por primera vez en 1986.

En la actualidad se encuentra disponible para las plataformas Windows, UNIX, Mac y Linux.

La programación en bloques constituye el pilar fundamental de LabVIEW, se diferencia de otros lenguajes de programación clásicos como C o Basic, ya que está basado en gráficos y no en texto como los otros, contando con una interfaz interactiva, didáctica y de fácil aprendizaje, ayudando a disminuir problemas relacionados con la sintaxis y la escritura de un lenguaje especifico de programación, la programación de bloques es una forma práctica y sutil de crear conciencia lógica en los futuros programadores.

## 2.3 Cuerpo De Un Programa En LabVIEW

Para nuestro primer ejercicio, haremos una operación básica entre dos números, para esto necesitamos de un panel frontal donde ingresaremos los dos números y a su vez, un indicador que nos mostrara el resultado de este programa. este ejercicio es un claro ejemplo de lo elemental que puede ser un ejercicio y como la lógica de programación, nos ayudará a desarrollarlo.

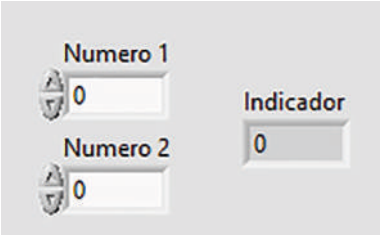



Figura 32. Panel frontal de la sumadora. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al crear los elementos en el panel frontal, estos aparecen automáticamente en el diagrama de bloques:

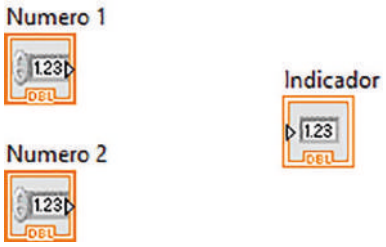



Figura 33. Diagrama de bloques de la sumadora. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En este punto, debemos recurrir a la lógica programadora, presentando una solución al problema evidenciado y buscando que, el programa realice la operación que nosotros deseamos hacer, se deben seleccionar las funciones dentro del diagrama de bloques, en este caso para una suma utilizaremos un código de programación como el siguiente:

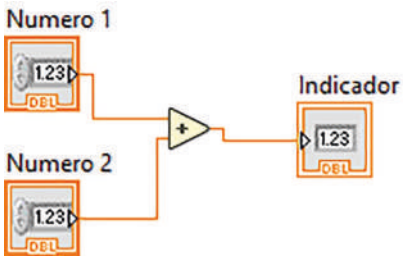



Figura 34. Conexión de los elementos del diagrama de bloques. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ya podemos ejecutar el programa, al no tener un ciclo while, tendremos que correr el programa continuamente, esto permitirá que el software tome los valores en tiempo real, ya que se estará ejecutando una y otra vez y presentando en el indicador, la sumatoria de ambos números.

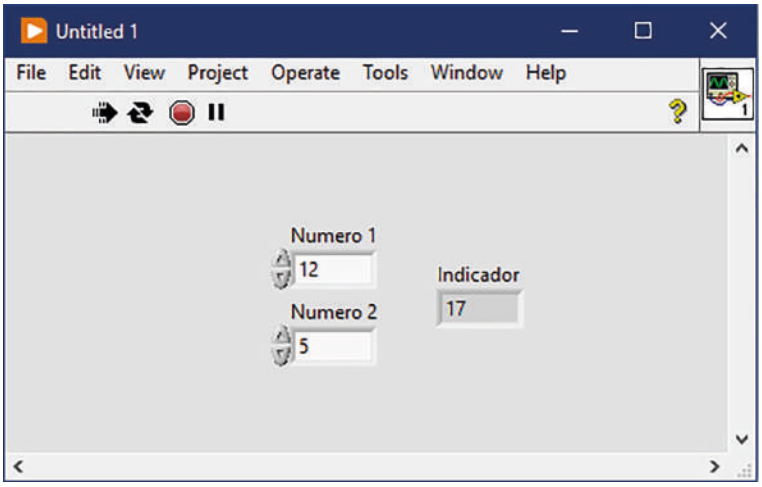



Figura 35. Programa en funcionamiento. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La interfaz del panel de control nos permite controlar las variables, actualizando los valores, por lo que, al cambiar un valor entre número 1 y número 2, obtendremos un resultado diferente en el indicador.

Para una resta, será tan sencillo como cambiar el operador matemático, ya sea eliminándolo o remplazándolo por el adecuado.

### 2.3.1 **Controles E Indicadores**

Los controles e indicadores de cada programa variaran en relación al tipo de proyecto que estemos realizando, basándonos en el ejercicio anterior podemos deducir que un elemento de control es aquel valor que puede ser modificado por el usuario, esto normalmente se hace a partir de perillas o flechas de incremento.

Ahora bien, los indicadores se diferencian porque no podrán ser modificados por el usuario, solo van a "indicar" algún valor o producto, teniendo en cuenta el ejercicio anterior el indicador sería el resultado entre la suma de ambos números

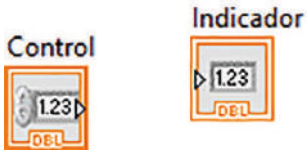



Figura 36. Diferencias entre indicador y control. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Dentro de los diagramas de bloques es fácil distinguir el control del indicador, el indicador tiene un borde delgado y una flecha a su lado izquierdo, por su parte, el control tiene un borde mucho más grueso y la flecha, está ubicada en el lateral derecho.

Si queremos cambiar, dentro del diagrama de bloques sin necesidad del panel de control, simplemente damos clic al bloque y lo cambiamos, ya sea de indicador a control o de control a indicador.

### 2.3.2 **Elementos booleanos**

Como sabemos, un elemento booleano funciona con un cero o un uno, verdadero o falso, no hay otro tipo de valor disponible.

Dentro de LabVIEW, los elementos se representan en letras, siendo la T, un valor 1 o verdadero, y la F, el valor de 0 o falso.

Como ejemplo, para encender un led con un switch, necesitamos un led conectado a un switch, para encender el valor, cambiando el estado del switch cambiará o encenderá el LED


Figura 37. Conexión de un switch y un led. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

De esta forma al presionar el switch se enciende el led.

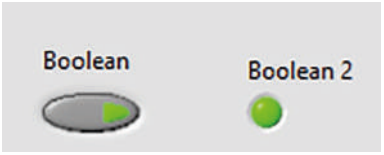



Figura 38. Ejemplo del uso de elementos booleanos.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 2.3.3 **Uso de Select**

Dentro del software, encontramos un menú para archivos de tipo comparación, con un icono denominado con Select, funciona bajo la estructura del condicional if.

Cuando la condición se cumple, devuelve un valor positivo, si no se cumple pues sencillamente no se envía nada.

Para evidenciar un Select, podemos apreciar un ejemplo tan sencillo como:

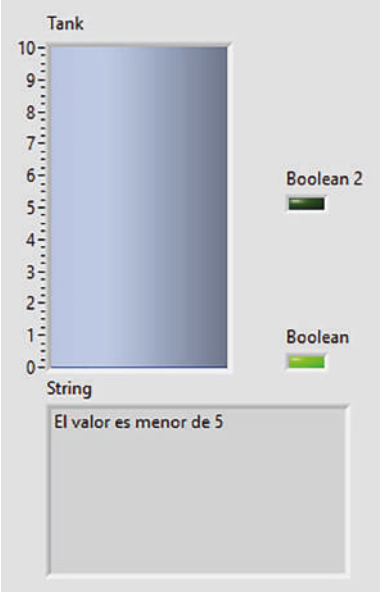



Figura 39. Panel frontal para ejemplo del uso de Select. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El objetivo se basa en encender un led, dependiendo del valor que tenga nuestro tanque, si es menor que 5, se encenderá el led de abajo, cuando supere este valor, se apagará el led de abajo y encenderá el led superior.

Utilizaremos dos Select, uno diferente en cada led.

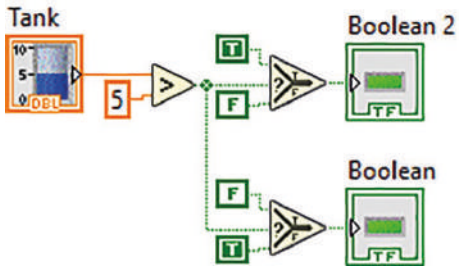



Figura 40. Diagrama de bloques del ejemplo del uso de Select. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Utilizando un control comparativo, en este caso, el mayor que, cuando el dato ingresado por el usuario sea mayor a 5, se encenderá el led superior, si es menor, se

encenderá el led inferior, Nótese que se utilizaron constantes booleanas para los Select, siendo efectivo al no utilizar ciclos.

Esta es una sencilla aplicación, sin embargo, la utilización de esta característica dependerá del programador y del ejercicio que desee realizar, siempre será un plus adicional contar con el desarrollo de ejercicios donde se puedan apreciar los detalles, detalles que hacen de un ejercicio o proyecto, un poco más completo.

### 2.3.4 **Uso De Cadenas De Texto**

Cuando queremos trabajar con texto en LabVIEW, usamos los controles e indicadores de texto, dentro del diagrama de bloques estos tienen un color rosa, junto a elementos booleanos y numéricos, los String también tienen constantes, indicadores y controles.

Complementando el ejemplo anterior, agregaremos cadena de texto para visualizar un tipo de texto en el programa, de tal forma que:

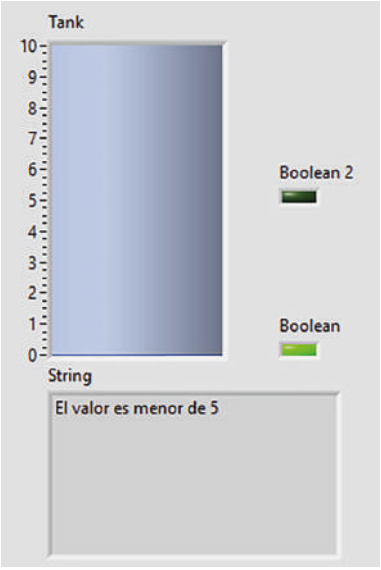



Figura 41. Ejemplo del uso de cadenas de texto. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La particularidad del indicador textual ex que podremos evidenciar que, si el valor es mayor o menor a cinco, no solo nos encienda un led, sino que también se nos muestre un mensaje indicando el estado

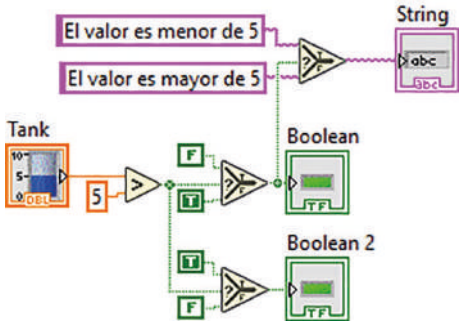



Figura 42. Diagrama de bloques del ejemplo de uso de String. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

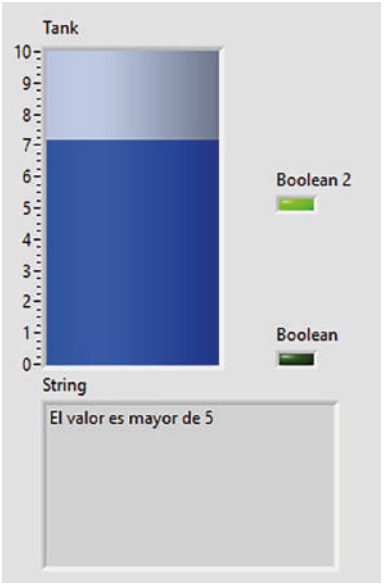



Figura 43. Programa funcionando. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 2.3.5 **Uso de la estructura Case**

La estructura case sirve para ejecutar posibles casos, el caso se selecciona con una variable de entrada, facilitando y disminuyendo la cantidad de condicionales

En el lenguaje de programación C, la sintaxis es:

```
switch (funcion)
{
    case1:
        Sentencias;
        break;
    case 2:
        Sentencias;
        break;
}
```

La estructura case, para el anterior ejemplo evalúa una variable y en funcion de esta, ejecuta una u otra sentencia diferente.

Dentro de un lenguaje gráfico, el control de las estructuras se representa mediante rectángulos, el case se conectará a cada opción y todas las operaciones serán almacenadas para cada caso independiente.

Para evidenciar el funcionamiento, partiremos del siguiente ejemplo:

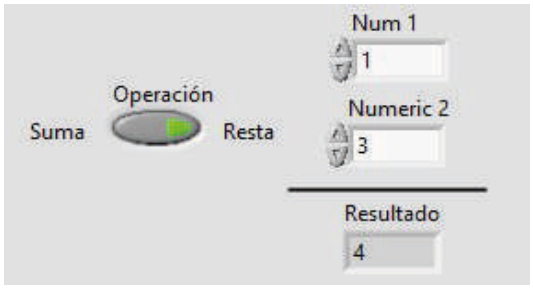



Figura 44. Ejemplo del uso de la estructura case. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Este pequeño programa, es una calculadora que tiene dos funciones, una para la suma y otra para la resta y se piden por dos números de entrada, el botón elige si estos números son sumados o restados.

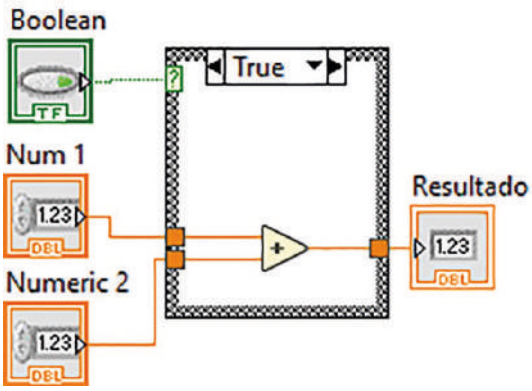



Figura 45. Diagrama de bloques del ejemplo de uso de case. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como mencionamos, el botón será el case, ya que si está en un estado u otro, podemos programarlo y adecuarlo como un case, si el slide está apagado, el valor será falso, si está encendido el valor será verdadero.

Al observar la Figura 45 notaremos inicialmente que lo único que hay para el case es una operación que se va a realizar, si por alguna razón metemos un control o indicador, solo servirán para un caso específico, para que esto funcione adecuadamente, se tendrán que crear variables de tipo local.

Por otro lado, dejando estos elementos afuera, y conectándolo al case podemos crear un cuadro donde cada valor de salida variará dependiendo del caso.

Para el case, se puede conectar un control tab, es útil para dar un aspecto mas estético, almacenando las operaciones y acciones en un contenedor individual, el tap control se encuentra dentro del menú de Containers.

Un ejemplo claro para calcular los valores de la ley de Ohm puede ser:

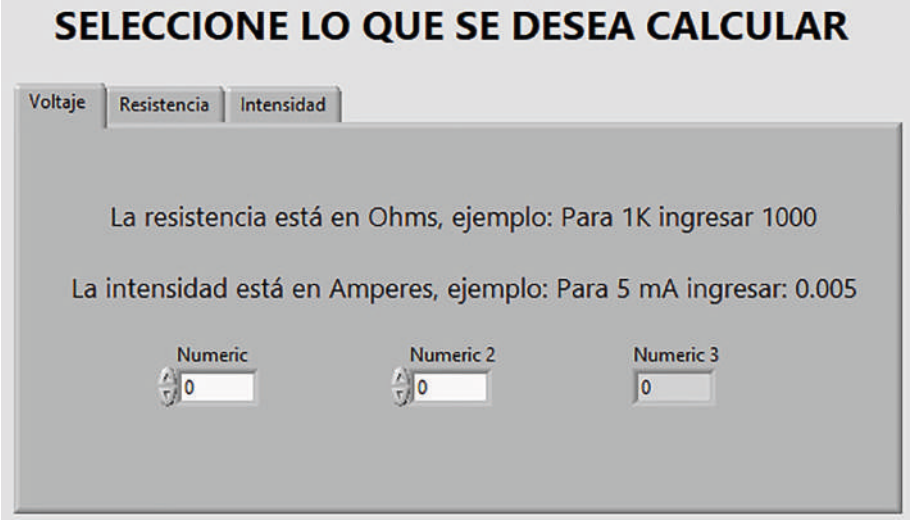



Figura 46. Ejemplo del uso de tab control y case.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Aquí evidenciamos tres pestañas, cada una de estas cambia en funcion del despeje, ya sea para obtener un voltaje, calcular una resistencia o una corriente.

Para cada pestaña, se cuenta con dos controles de tipo numero y un indicador, ingresando los dos valores, se calculará el tercero mediante funciones matematicas2.

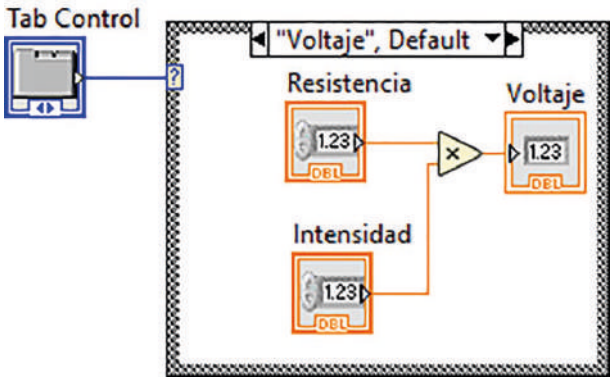



Figura 47. Programa para el uso del TAB control. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

De la figura 47, podemos apreciar que se meten los elementos en cada caso, se utiliza un control tab y resulta mejor crear controles e indicadores en cada paso, para la ley de ohm se hacen las respectivas operaciones matemáticas, multiplicación y división.

Es importante destacar que, al conectar el control al case, automáticamente aparecerán los primeros casos, para agregar más casos solo se debe dar clic derecho al nombre del caso y presionar en add case before o after.

Si necesitamos cadenas de texto, dentro de un case conectado a un botón, también lo podemos hacer siguiendo el ejemplo que se muestra a continuación.

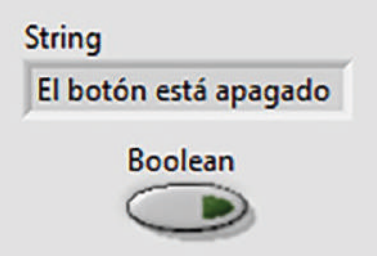



Figura 48. Ejemplo del uso de String con case. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El diagrama de bloques quedará como

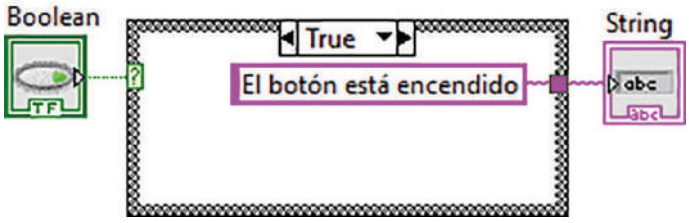



Figura 49. ejemplo de String en case. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la situación presente, si el programa se ejecuta de manera constante, es probable que se presente el mensaje correspondiente al caso false, puesto que es probable que

el botón esté en estado false al momento de la ejecución del programa. No obstante, al activar el botón, el mensaje correspondiente al caso true será mostrado.

### 2.3.6 **Uso de Array**

El conjunto de datos conocido como array (también llamado matriz o vector) se utiliza para almacenar múltiples elementos del mismo tipo. Es importante tener en cuenta que los Arrays no pueden contener objetos de distintos tipos. Por ejemplo, si un array contiene leds, solo puede contener elementos que sean del tipo led.

En LabVIEW, es común que se requiera utilizar los elementos de un array de forma individual. Para lograr esto, se pueden utilizar diversas herramientas disponibles en el diagrama de bloques, como el índex array y el initialize array.

Cuando se agrega un array al diagrama de bloques en LabVIEW, este se ajusta automáticamente al tamaño del objeto que se arrastra dentro de él. Si es necesario aumentar el tamaño del array para almacenar más elementos del mismo tipo, se puede hacer clic en la flecha de tamaño que se encuentra en la parte inferior o superior del array y desplazarla hacia el lado correspondiente. Si se requiere que el array sea de dos dimensiones, se le da clic derecho y se da clic en add dimension.

Un ejemplo del uso de Arrays puede ser el siguiente:

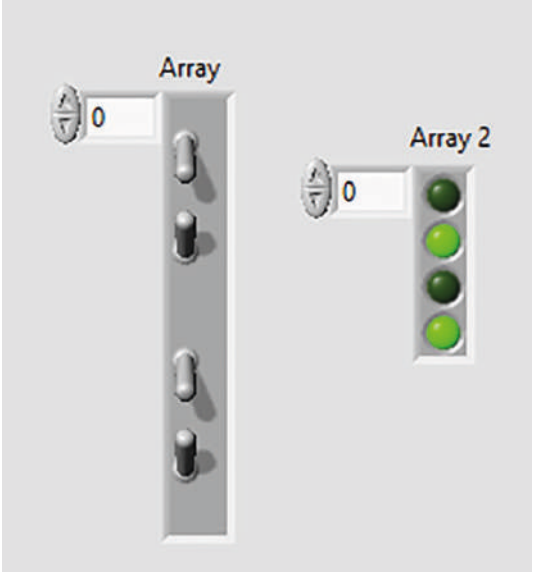



Figura 50. Uso de Array en LabVIEW.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En el panel frontal, se tiene el array para los Leds y otro para el switch. Autoría Propia.

Para el diagrama de bloques, solo tendremos dos elementos que son los Arrays, si los conectamos directamente el programa funcionara a la perfección.


Figura 51. Diagrama de bloques del ejemplo de Arrays. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Esto ayuda a disminuir el código, ya que, si no se usan Arrays, en el diagrama de bloques debería haber cuatro leds y cuatro switches, por lo que tendríamos 8 bloques, de esta forma solo debemos tener dos bloques.

Si se quiere controlar los cuatro leds con un solo switch será un poco complejo, ya que estaremos tratando de controlar los leds con un distinto tipo de elemento.

Si conectamos en este orden, los Arrays respetarán su orden, el led 1 se encenderá con el switch 1 y así sucesivamente, con la herramienta índex array podremos invertir el sentido de los leds y del switch, encendiendo el led 1 con el switch 4

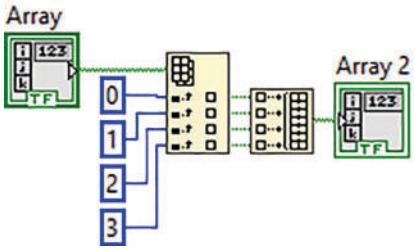



Figura 52. Uso de Índex array y Build array. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En base a la figura 52, se definió un orden ascendente de elementos de switches, de tal forma que el build array se conecto al array de leds.

El programa debería funcionar adecuadamente

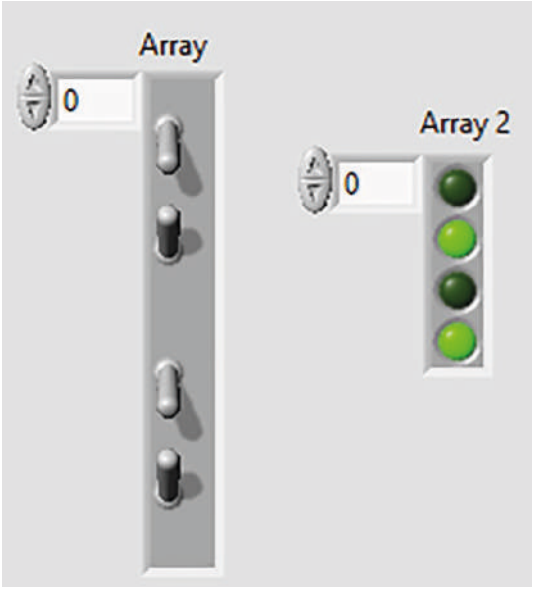



Figura 53. Funcionamiento del programa con Índex array.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 2.3.7 **Uso De Arrays Con String**

Para las cadenas de texto que deseamos mostrar como mensajes, dentro de los elementos que están los Arrays, podemos utilizar el ejemplo anterior y mostrar lo siguiente.

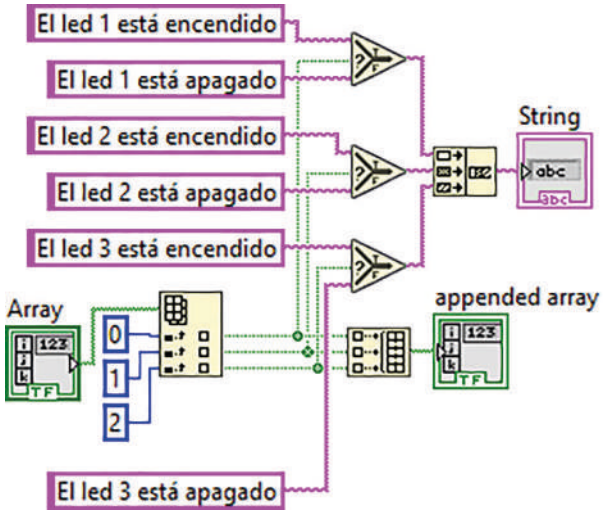



Figura 54. Uso de String y array. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

De la figura 54, el array del switch se indexa, con el fin de sacar los elementos y ordenarlo y así conectar cada elemento a un Select, evaluando de tal forma si está o no, activo el switch.

Si es falso, con una constante de texto se dice que el led está apagado, si es verdadero, dice que está encendido.

Es interesante evidenciar que las tres salidas de texto van a un indicador, simplemente se usa la herramienta de concatenar String, lo cual agrupa todas las constantes de texto y saca una sola salida, construyendo el array y conectándolo al indicador.

El resultado debería ser el siguiente:

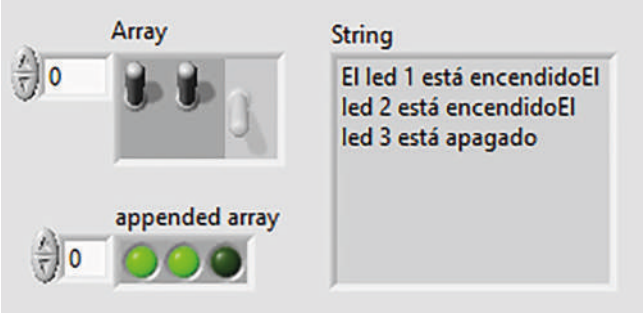



Figura 55. Programa funcionando con Concatenate String. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como podemos apreciar, los textos del indicador se ven un poco amontonados, simplemente se agrega un salto de línea utilizando un Carriage return constant, de tal manera que:

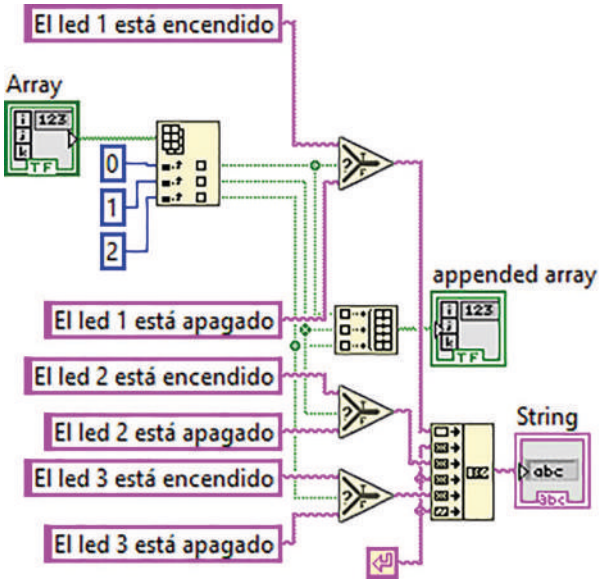



Figura 56. Uso de Carriage return constant. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El resultado quedará así:

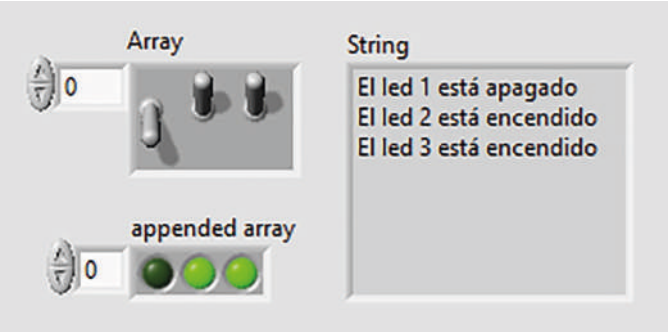



Figura 57. Programa funcionando con Carriage return constant. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 2.3.8 **El Ciclo For**

El ciclo for es una estructura de control ampliamente utilizada en la mayoría de los lenguajes de programación. Esta estructura de control permite establecer el número mínimo de iteraciones requeridas para que un conjunto de instrucciones se repita.

Los elementos básicos que conforman un ciclo for son: la variable de control, la inicialización de la variable de control, la condición de control, el incremento y el cuerpo del ciclo.

La variable de control es el elemento central del ciclo for, ya que es la variable que el bucle utiliza para trabajar. Se inicializa para establecer un valor predeterminado con el que iniciará la iteración. Luego, durante cada iteración, el valor de la variable de control se actualiza de acuerdo con el incremento especificado en el bucle.

El cuerpo del ciclo for contiene el conjunto de instrucciones que se deben ejecutar en cada iteración. Este conjunto de instrucciones se repetirá hasta que se cumpla la condición de control, momento en el cual el ciclo for finalizará.

Su uso se orienta a vectores, permite agregar, modificar o eliminar datos según el índice.

En LabVIEW, este ciclo también se representa con un rectángulo:

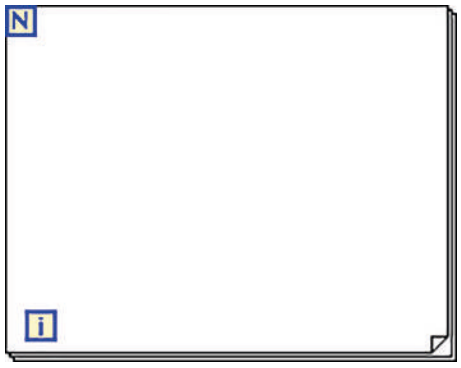



Figura 58. Representación gráfica del ciclo for. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En el valor de N, se representará el número de veces que se piensa repetir el ciclo, la terminal i representa la iteración o cambio que va desde 0 hasta N-1

Dentro de este ciclo es importante agregar un timing, delay o waiter, con el fin de que no se consuma toda la memoria o se sature el sistema, para el delay, el valor mínimo es de 5 milisegundos.

Podemos tomar el siguiente programa como ejemplo del uso del ciclo for:


Figura 59. Ejemplo del uso del ciclo for. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En el panel frontal solo tenemos un indicador numérico.

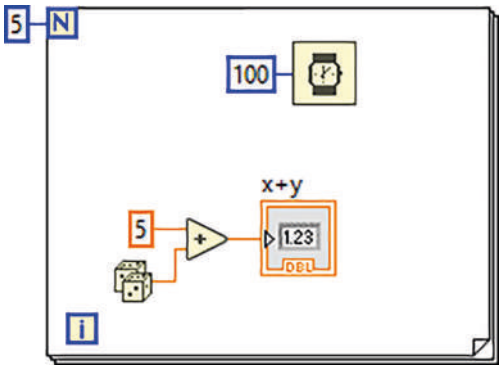



Figura 60. Ejemplo del ciclo for. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El número de iteraciones del ciclo es de 5, la operación estará dada por una suma con valores aleatorios y un cinco, resultado que, posteriormente se evidenciará en el indicador numérico.

La función de Timer ya que servirá para ir apreciando los resultados de forma secuencial, si este valor es bajo, prácticamente veremos solo el ultimo ciclo realizado y el ultimo valor calculado.

Si quisiéramos que el numero de iteraciones quede en un numero al azar, en vez de 5 sea un numero N, entonces podemos conectar un Random a la terminal N del ciclo for.

### 2.3.9 **For Condicional**

Un ciclo for puede considerarse como condicional cuando se le agrega una terminal de paro. Esta terminal permite conectar un botón de detener o una condición que detenga el programa en un momento determinado, incluso si no se han completado todas las iteraciones establecidas.

Para agregar una terminal de paro a un ciclo for en LabVIEW, se puede hacer clic derecho sobre el ciclo for y seleccionar la opción "Conditional Terminal". Al agregar esta terminal, se puede establecer una condición que indique cuándo se debe detener el ciclo for. Por ejemplo, se puede establecer una condición que detenga el ciclo si se

cumple cierta condición o si se presiona un botón de detener específico. De esta manera, se puede detener el ciclo en cualquier momento que sea necesario.

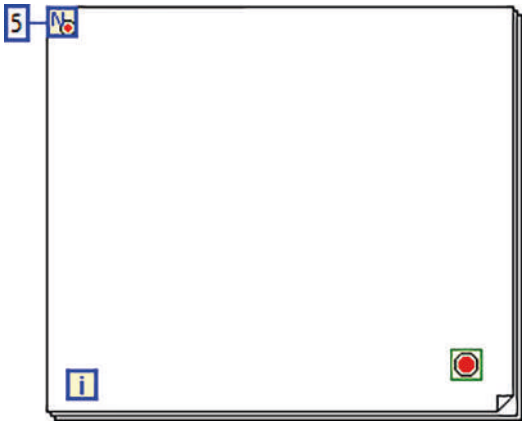



Figura 61. Representación del ciclo for. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Nótese que en la terminal N aparece un cuadro verde con un circulo rojo, en la esquina superior derecha, esto servirá para usarlo como una señal de paro

### 2.3.10 **Shift Register**

El shift Register es un elemento que guarda valores y mediante un tipo de operaciones, permite modificarlos, siendo útil en una gran variedad de programas, más en esos donde los contadores son comunes.

Estos elementos se utilizan en las repeticiones, para agregar uno, se debe dar clic en el ciclo y posteriormente dar clic en add shift Register.


Figura 62. Representación gráfica del shift Register en ciclo for. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La terminal de la izquierda es el valor iniciar donde se guardará el valor del shift Register, mientras que la terminal lateral derecha, se guardará el nuevo valor

Un ejemplo del uso es el siguiente:

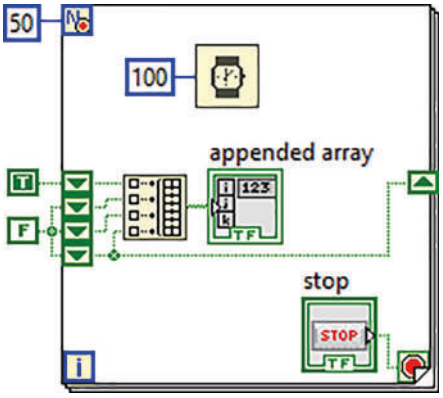



Figura 63. Ejemplo del uso de shift Register y array. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En este ejemplo, contamos con 4 leds y un shift Register, aumentando a 4 elementos que representa cada led, esto se hace colocando el curso bajo el shift Register y arrastrándolo hacia abajo.

En este caso, inicializamos una constante booleana, que indica que el primer led se encuentra encendido y al ejecutar el programa, en un lapso de 100 milisegundos pasa a un estado de false y el siguiente será true, así hasta llegar al ultimo y esto mientras se repite 50 veces o hasta que se presione el botón stop.


Figura 64. Panel frontal del programa de ejemplo de shift Register y array. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 2.3.11 **El Ciclo While**

A diferencia del ciclo for, el ciclo while ejecuta una determinada acción hasta que se cumpla una condición especificada o hasta que el programa finalice su función. Si el programa no tiene un final determinado, el ciclo while puede ejecutarse indefinidamente, lo que puede causar problemas de rendimiento y bloquear el programa. Por esta razón, es importante tener una condición de paro adecuada para evitar problemas.

En el ciclo while, es necesario especificar una condición de paro en la terminal de stop para que el ciclo funcione correctamente. Esta condición de paro puede ser una variable o una expresión lógica que indique cuándo debe detenerse el ciclo. Si no se especifica una condición de paro, el ciclo while se ejecutará continuamente, lo que puede agotar los recursos del sistema y causar problemas de estabilidad.

Figura 65. Representación gráfica del ciclo while. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En este caso también se puede utilizar el shift Register.

A continuación, se puede apreciar un ejemplo sencillo para calcular la hipotenusa de un triangulo basado en el teorema de Pitágoras, con el ciclo while.

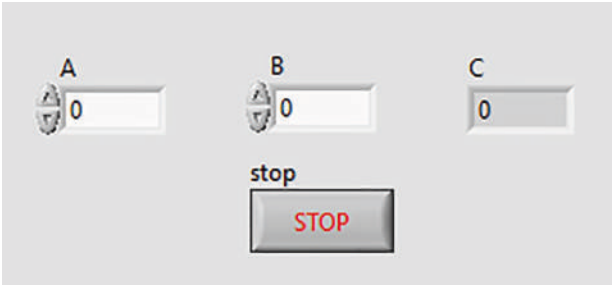



Figura 66. Ejemplo del uso del ciclo while. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El panel frontal cuenta con dos controles numéricos y un indicador, cada control representa un lado del triangulo y el indicar representa la hipotenusa.

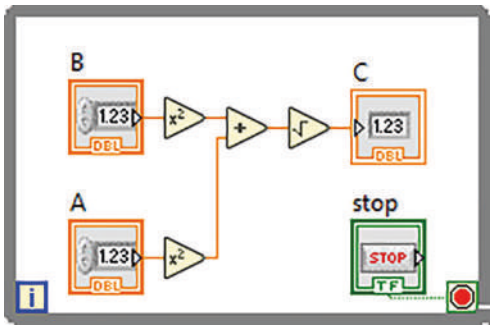



Figura 67. Diagrama de bloques del ejemplo con while. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para el diagrama de bloques, se debe usar un ciclo while para contener toda la operación, partiendo del teorema de Pitágoras, el cálculo de la hipotenusa está dada por:

$$c^2 = a^2 + b^2$$

$$c = \sqrt{a^2 + b^2}$$

La manera más sencilla para dar solución al ejercicio, es utilizando el operador matemático square, elevando al cuadrado cualquier numero que se le conecte, por lo que usamos dos, sumamos los resultados y posteriormente, sacamos la raíz cuadrada.

El valor del resultado, se evidencia en el indicador.

Nótese que el ciclo while tiene un botón de stop conectado a la terminal de paro, sin esto, el programa no se ejecuta.

Para evitar problemas en un futuro o con proyectos con más elementos, debemos colocar siempre un Wait.

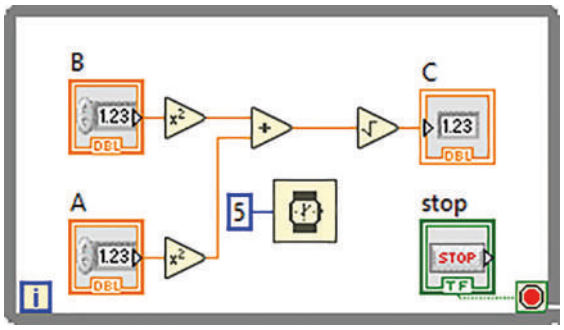



Figura 68. Uso del ciclo while con Timing. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Un ejemplo para el uso de shift Register con un ciclo while, podría ser un contador, para llenar un tanque.

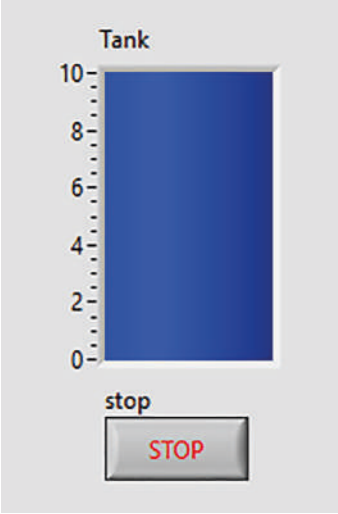



Figura 69. Panel frontal para el programa con while y shift Register. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ahora bien, el objetivo del proyecto será que el tanque se vaya llenando como si se tratara de una maquina automática de algún líquido.

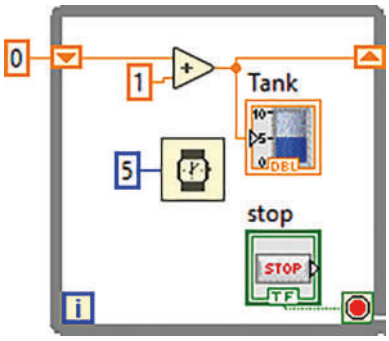



Figura 70. Diagrama de bloques del ejemplo de un contador. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El shift Register se debe inicializar en 0 y su incremento será en uno, esto hace que, durante cada repetición, el valor se sume en uno, al valor inicial, esto irá incrementando lentamente.

El timer es importante, ya que, si se deja sin un retraso, el llenado va a ser tan rápido que no se podrá apreciar la variación progresiva del llenado del tanque.

Si queremos aumentar la presión del llenado del tanque, podemos utilizar decimales y condicionales y obtendremos un mejor resultado.

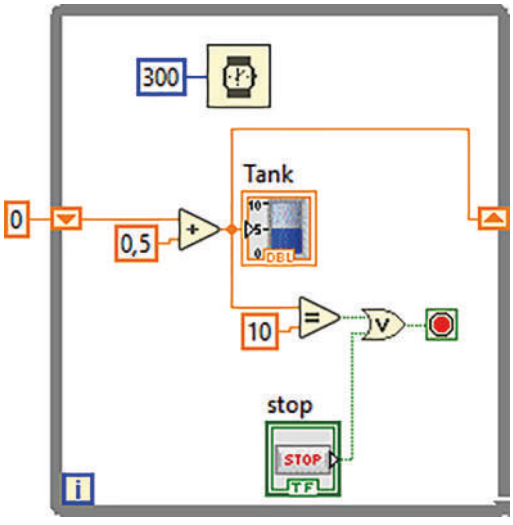



Figura 71. Diagrama de bloques del ejemplo terminado. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 2.3.12 **Variables Locales**

Dentro de los lenguajes de programación como C o Java, encontramos distintos tipos de variables, las variables locales suelen ser utilizadas dentro de funciones pequeñas, por otro lado, y dependiendo de la situación, las variables cambiarán por globales.

En LabVIEW, podremos hacer lo mismo, crenado variables locales y usarlas varias veces en distintas partes del código, claro está que esto no es recomendable.

Para crear una variable local, solo se da clic derecho sobre el elemento, luego en create>Local variable.


Figura 72. Control numérico slide y su variable local. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La variable local parece una constante, si se le da un clic, se puede cambiar por otro elemento que se encuentre en el diagrama de bloques.

### 2.3.13 **Uso de Cluster**

En LabVIEW un Cluster es una colección de elementos de diferentes tipos. Es parecido al array, la diferencia es que en el array sólo se pueden introducir un solo tipo de elementos, es decir; un led o un botón. En cambio, en el Cluster se puede introducir cualquier elemento, ya sean leds, botones, medidores, controles numéricos y de texto, etc.

Un Cluster bien programado, podrá disminuir evidentemente el tamaño del código, siendo una herramienta totalmente imprescindible

Podemos ver un panel frontal con un Cluster que alberga algunos elementos:

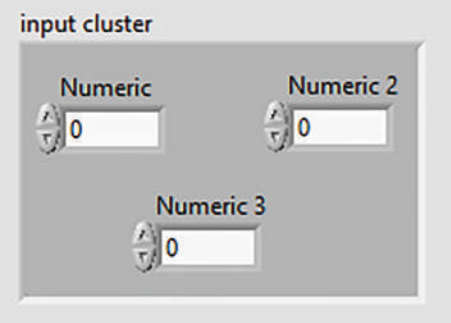



Figura 73. Cluster con un led, un botón y un indicador de texto. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como podemos apreciar en la figura 73, la gráfica evidencia tres elementos diferentes, donde el tamaño podrá ser modificado por el programador.

Disminuyendo el código, el diagrama para un Cluster solo será un bloque, por lo que reduce sustancialmente el trabajo del programador y la visualización del código.

Para sacar los elementos del Cluster, podemos usar Unbundle y Unbundle by name. La diferencia entre estos es que el Unbundle saca los elementos de forma individual, en orden ascendente. Y el Unbundle by name saca los elementos mostrando sus nombres, en algunas ocasiones es más fácil utilizar esta función para saber con qué elementos se está trabajando.

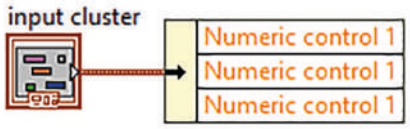



Figura 74. Unbundle by name. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Si unimos el Unbundle by name al cluster, se mostrará un elemento y si se requiere visualizar más elementos solo se debe colocar el cursor debajo de la función, al dar clic se mostrarán los siguientes elementos.

Se puede observar el siguiente ejemplo donde se usa un Cluster y variables locales:

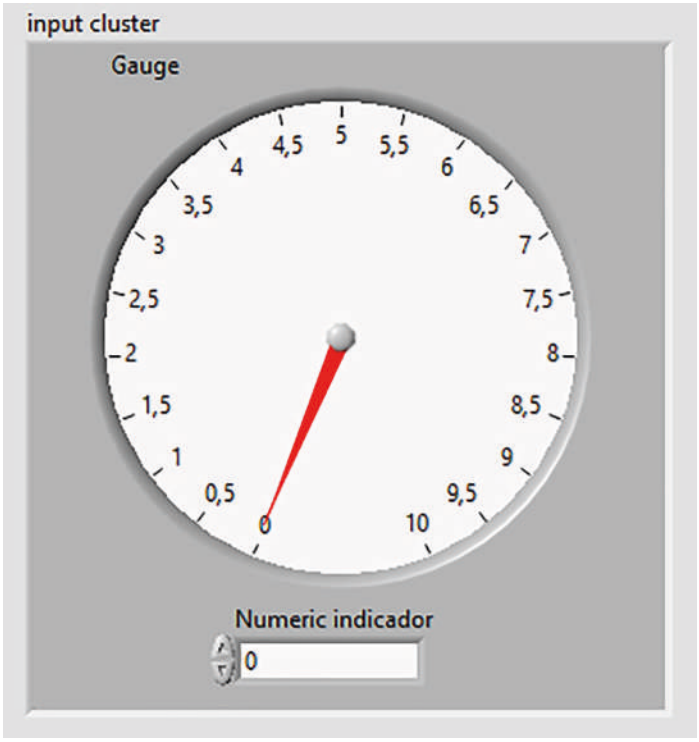



Figura 75. Ejemplo de Cluster. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como se puede apreciar de la figura 75, el panel frontal consta de un medidor y un indicador, se pretende que, al moverse la aguja del medidor, el valor pueda ser apreciado en el indicador numérico.

Inicialmente, procedemos a sacar los elementos de cluster, por medio del Unbundle se selecciona el medidor que usaremos para el control, por medio de Bundle by name, conectaremos el medidor al indicador y aquí, prácticamente ya estaríamos programando un cluster.


Figura 76. Uso de Cluster con variable local. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

De esta manera, al correr el programa y mover la aguja del medidor, el indicador numérico nos mostrará el número real que la aguja está marcando.

## 2.4 Práctica #1: Maquina De Refrescos

Con el desarrollo de las anteriores practicas procederemos a crear un programa pedagógico, simulando una máquina expendedora de líquido, como los que podríamos encontrar en las cadenas reconocidas de comidas rápidas.

Para el procedimiento, gráficamente mostraremos tres botones, dichos botones podrán representar un sabor distinto, como puede ser el sabor a Cola, Manzana y Limón, por ejemplo. Dicha interfaz mostrara un mensaje de bienvenida mientras no se esté utilizando la máquina.

El control numérico representa el lugar donde se introduce el dinero junto con el indicador que imprimirá el cambio

Adicionalmente, habrá un indicador que irá simulando el llenado del vaso, siempre y cuando se cumpla la condición de haber introducido una cantidad de dinero suficiente.

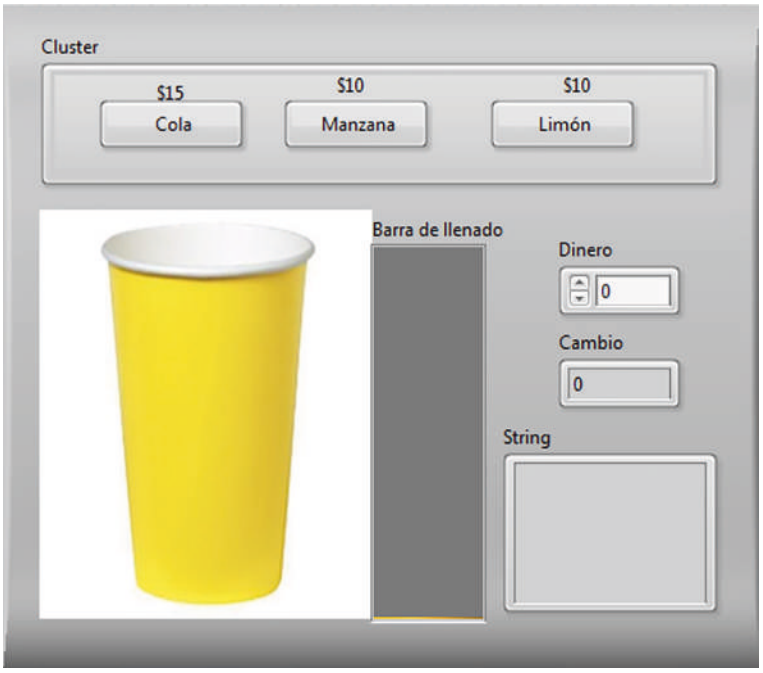



Figura 77. Panel frontal de refrescos. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Los botones que indican los sabores están dentro de un Cluster, lo demás son controles e indicadores.

Entonces, al correr el programa, se mostrará un mensaje de bienvenida y el usuario podrá ingresar dinero y presionar el botón que elija.

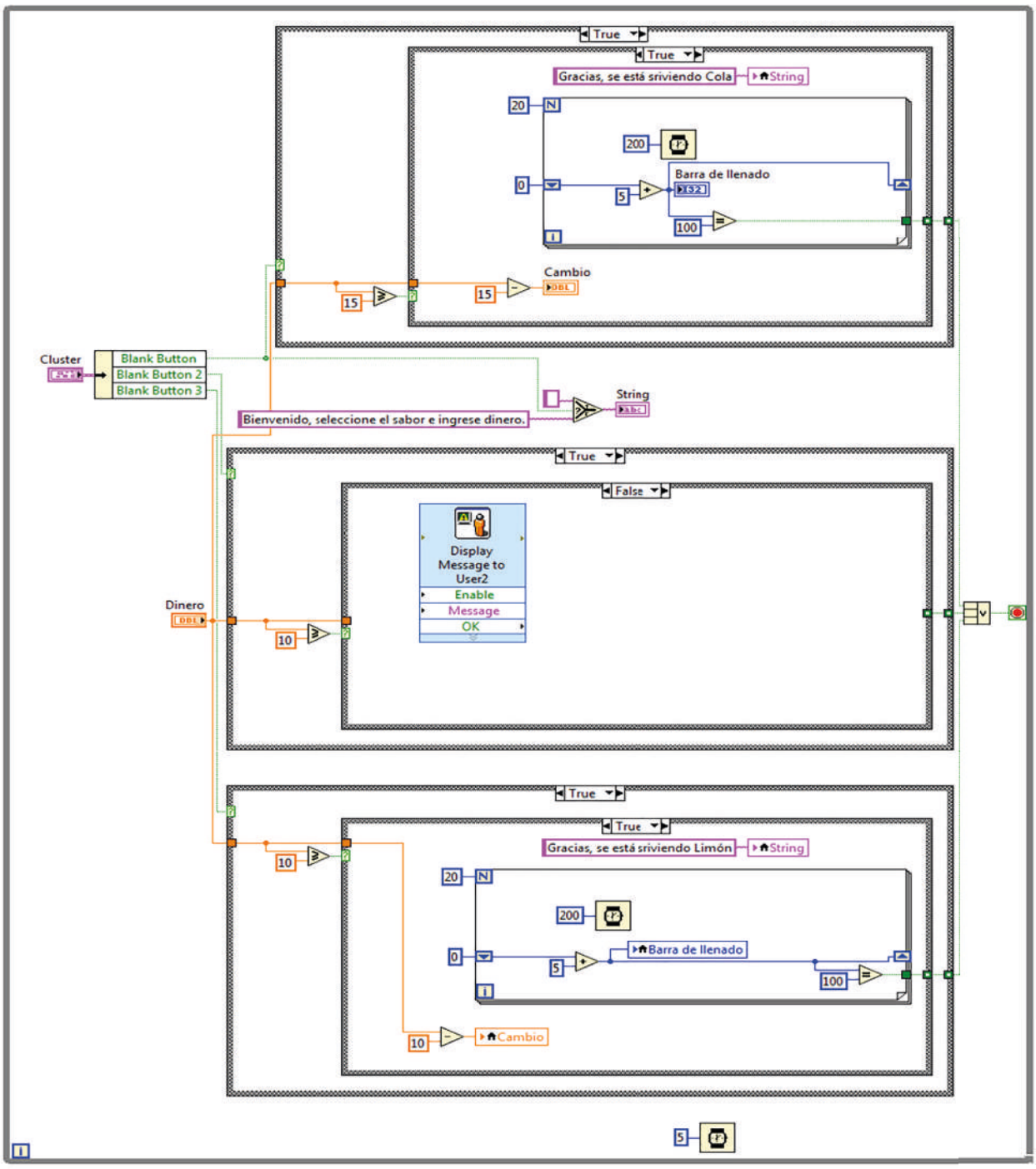



Figura 78. Diagrama de bloques de la máquina de refrescos. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como se puede ver en la Figura 78, se utiliza el while que une a 3 estructuras de tipo case, anidadas a un ciclo for.

Analizaremos cada parte del código.

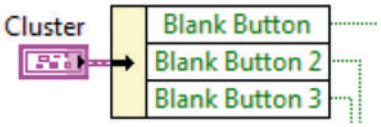



Figura 79. Se sacaron los elementos del Cluster con Unbundle by name. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Dentro del while, se utilizó Unbundle by name para poder usar los botones del Cluster individualmente. Estos tres botones se conectaron a una estructura case cada uno.

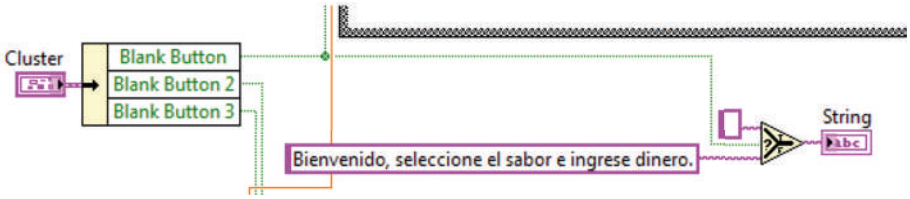



Figura 80. Mensaje de bienvenida del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El mensaje de bienvenida, no debería estar dentro de ningún caso, ya que este se debe mostrar al momento de ejecutar el programa, por medio de un Select se conecta nuestro mensaje a cualquier caso, por lo que, al activar cualquier botón, lógicamente efectuará una acción por lo que el mensaje deberá quitarse de la interfaz.

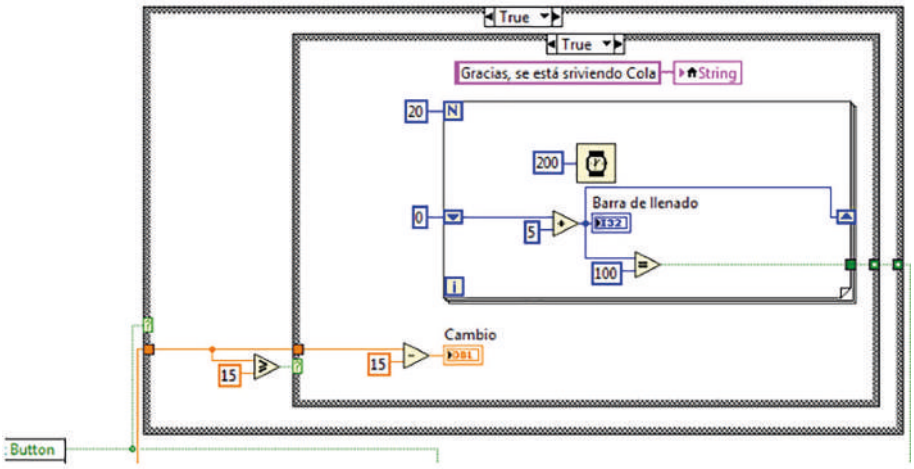



Figura 81. Case para el primer botón Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Cada case tiene un mismo botón y un funcionamiento similar, cambiará únicamente el costo de la bebida y el nombre, cuando el caso se encuentra en el valor true, se determina si el valor ingresando de dinero es mayor que el costo de la bebida.

Durante este proceso, se mostrará el mensaje de que la bebida esta siendo servida y se utilizará como una variable local, para el indicador del texto

Para simular el llenado del vas, utilizamos un ciclo for y un delay con, con un shift Register ira aumentado el contador a medida que se llena el vaso, empezando desde 0 y con un incremento de 5 cada 200ms, cuando el valor del tanque sea igual a 100, se parara el programa.

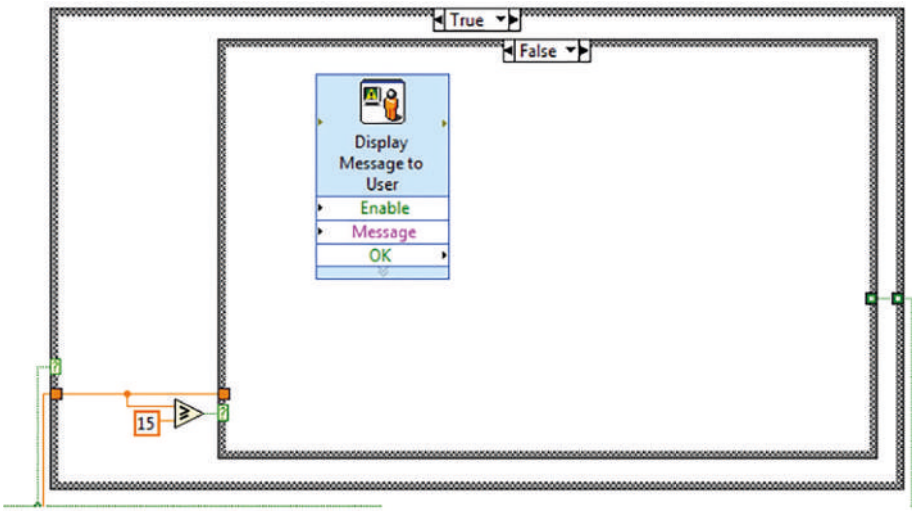



Figura 82. Caso 2, valor falso. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Condición: cuando el valor ingresado sea menor a 15, el programa mostrara el mensaje de que el dinero no alcanza para realizar la compra.

Los siguientes dos case indican lo mismo, sólo cambia el valor del dinero y que se usan variables locales de los controles e indicadores.

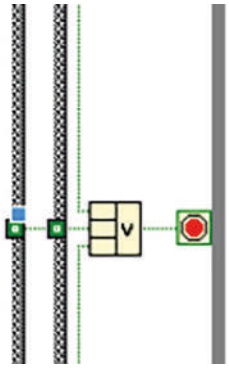


Figura 83. Uso de compuerta OR. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La compuerta OR de tres casillas permite detener el programa cuando cualquiera de los tres tanques pertenecientes a cada botón, llegue a 100.

## 2.5 Inserción De Imágenes Y Decoraciones

Del ejercicio anterior, podemos apreciar que tenemos imágenes que evidentemente, dentro de nuestro software no aparecen, para esta sección se mostrará como se insertan imágenes, para esto se requiere tener la imagen almacenada en nuestro computador y la implementación en el software es tan sencilla como arrastrarla desde su ubicación o carpeta hasta el panel frontal.

Para encontrar las decoraciones, nos vamos al menú Modern > Decorations.
Ahí podemos encontrar el siguiente menú:

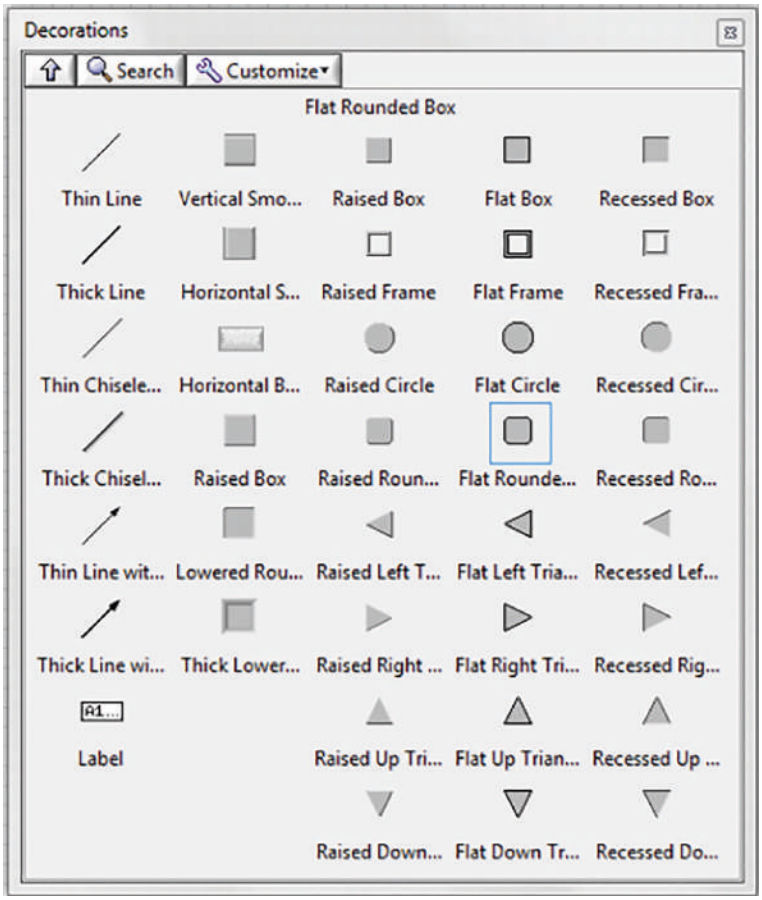



Figura 84. Menú de decoraciones. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En este menú, podremos elegir una serie de objetos decorativos que harán de nuestra interfaz gráfica, una interfaz más sólida y más didáctica.

Para que la decoración que hayamos elegido no tape todo, nos vamos al menú y damos clic en el ícono que tiene dos flechas circulares:

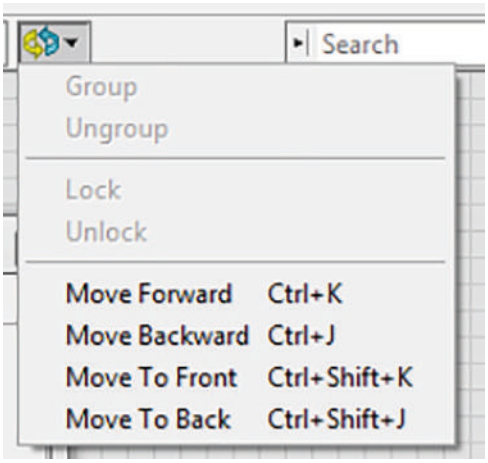



Figura 85. Menú para mover elementos. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En esta opción, seleccionamos Move to Back, con esto haremos que nuestra decoración se quede por detrás de todo lo que haya en el panel.

Para insertar imágenes al panel frontal es muy sencillo, solo basta con dar clic en Edit o Editar, y en Import picture to clipboard.
De esta manera se puede elegir la imagen que se requiera.

Otra forma es arrastrar la imagen desde su ubicación al panel frontal. Cualquiera de estas dos opciones es bueno, y luego se puede modificar el tamaño de la imagen en el panel frontal.

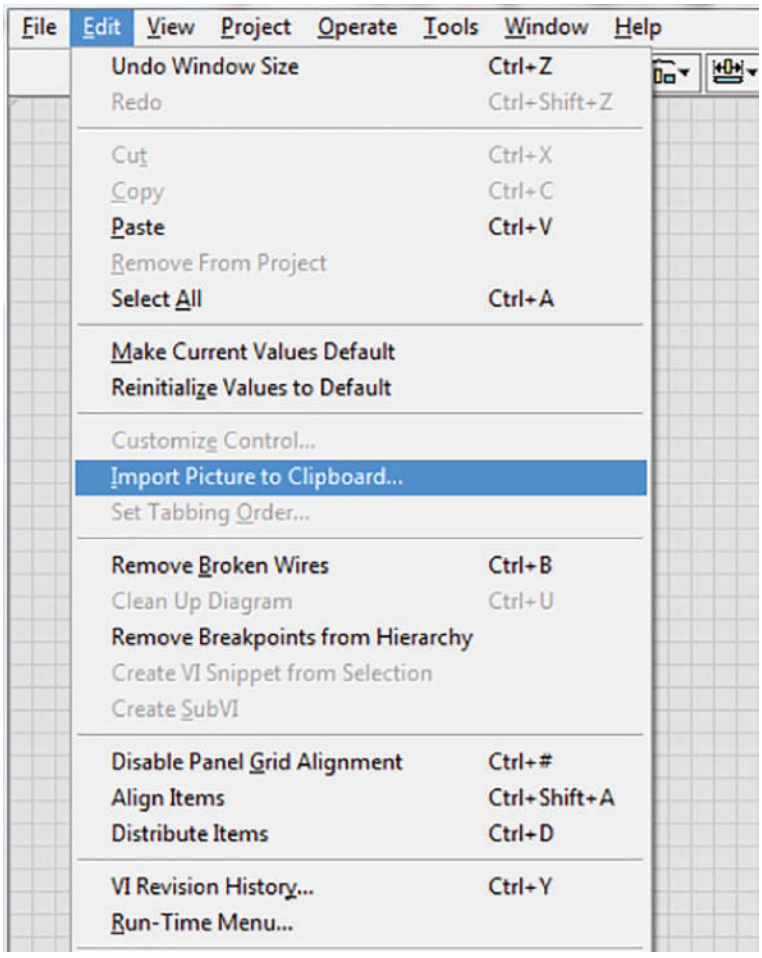



Figura 86. Insertar imágenes en el panel frontal. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

## 2.6 Máquinas De Estados

Una máquina de estados se usa para el desarrollo de algoritmos, siendo una forma muy eficaz de implementación.

Se sigue una serie de pasos para lograr que la máquina funcione, esto se puede observar en el siguiente diagrama:

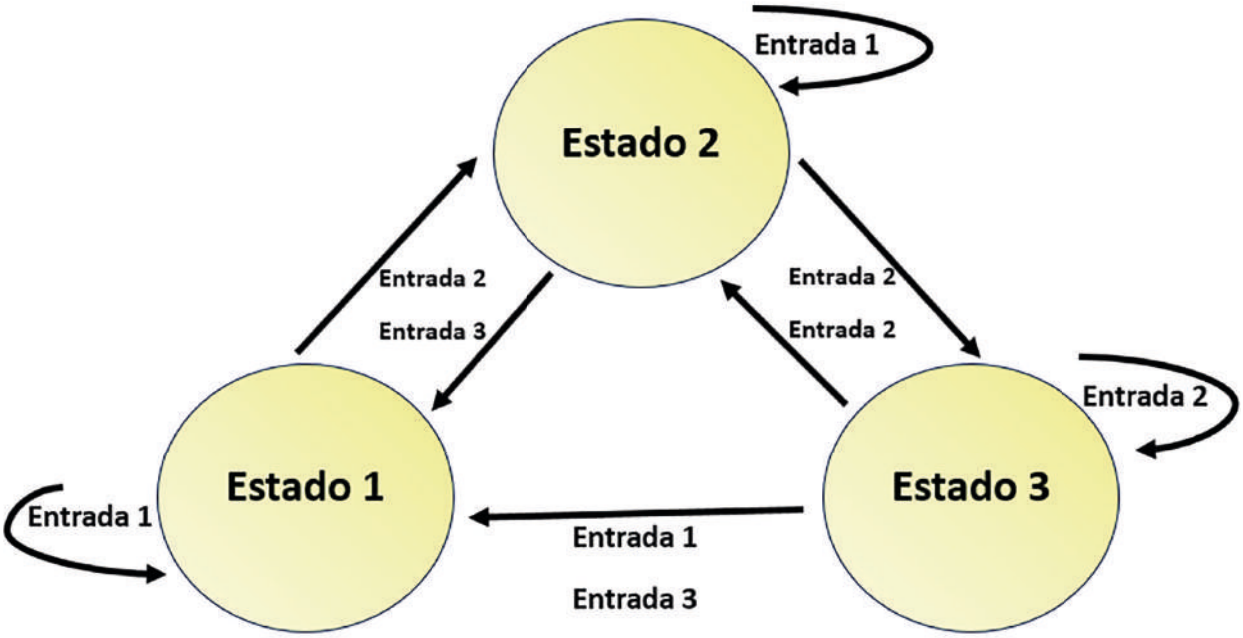



Figura 87. Diagrama de estados. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Las maquinas de estado, en LabVIEW, se utilizan para resolver distintos problemas y condiciones determinantes.

Los elementos de una maquinan de estados son: una estructura case, un ciclo while y los elementos básicos de las estructuras, ya se aun timer o un botón para el paro del ciclo while.

La plantilla de una maquina de estados en LabVIEW puede abrirse si entramos a la siguiente dirección:

File > new > For template > Frameworks > Designs Patterns > Estándar state machine.

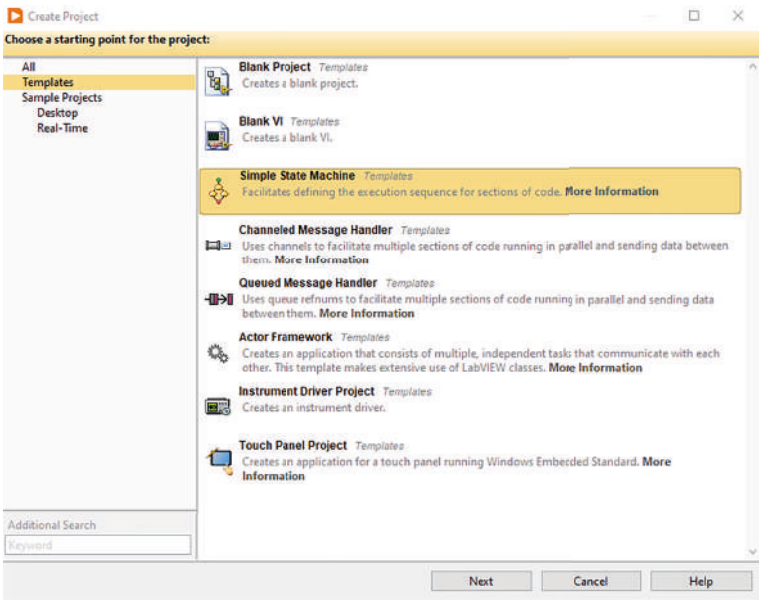



Figura 88. Proceso de apertura de plantilla de máquina de estados. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Si presionamos en OK, se abre dentro de la interfaz del diagrama de bloques una plantilla con las maquinas de estados, la estructura case añade estados y controlan la variable con un shift Register.

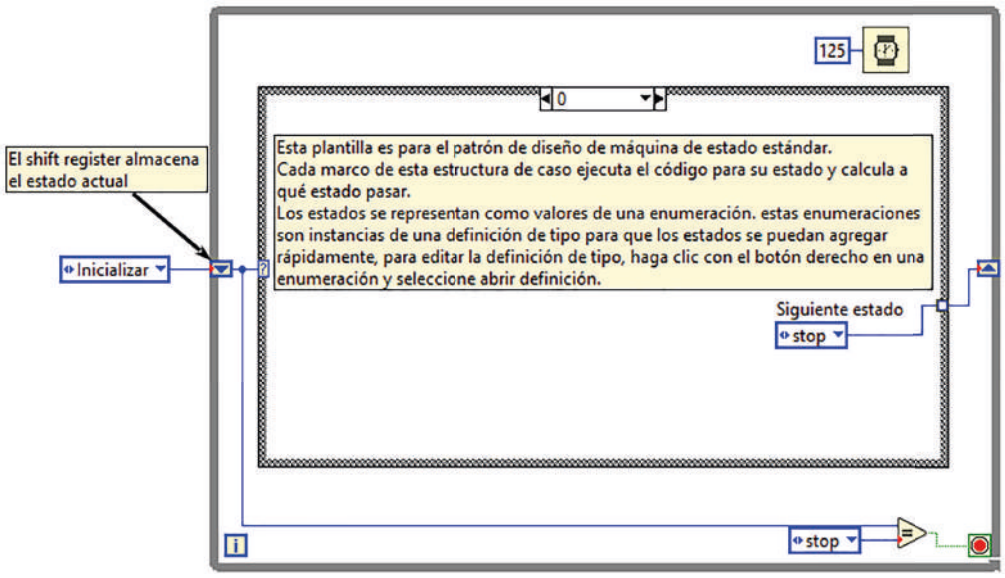



Figura 89. Plantilla de máquina de estados en LabVIEW. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para el uso de una máquina de estados, es necesario crear un programa de llenado de tanque, el panel frontal muestra que el tanque tiene una capacidad de 1000 litros y 3 botones, junto con el indicador numérico

Para empezar, debemos establecer nuestros estados, salidas, entradas, acciones para poder realizar el diagrama de estados.

Para este programa, tenemos lo siguiente:

Estados:

- Stand by: El programa en estado normal
- Inicio: Se comienza a llenar el tanque
- Detener: Se detiene el proceso de llenado

Entradas:

- Botón de inicio
- Botón de detener

Salidas:

- Llenado del tanque
- Paro del programa

Acciones:

- Si se está Stand by se espera una acción.
- Si se está en inicio se llena el tanque hasta llegar a 1000.
- Si se está en detener, se detiene el proceso de llenado, pero no la ejecución del programa.

Con estos datos ya podemos crear el diagrama de estados que nos servirá para poder desarrollar el código del programa en LabVIEW.

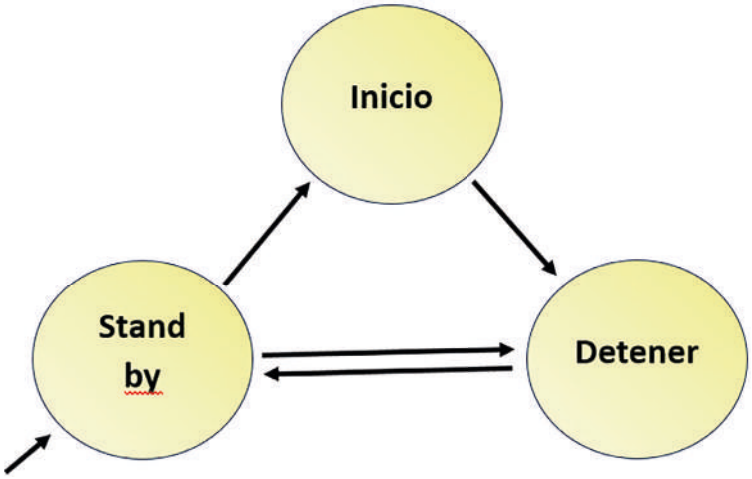



Figura 90. Diagrama de estados general. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ahora, ya podemos empezar con el código. Colocamos lo básico, un ciclo while, un case y un Enum.

En el panel frontal evidenciamos un tanque, dos botones que serán usados para iniciar el proceso y detenerlo y, por último, el indicador numérico para visualizar la cantidad de litros.

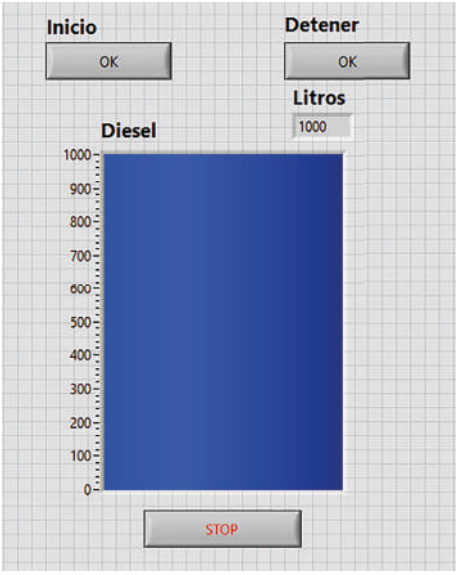



Figura 91. Panel frontal del ejemplo de máquina de estados. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El diagrama quedará así

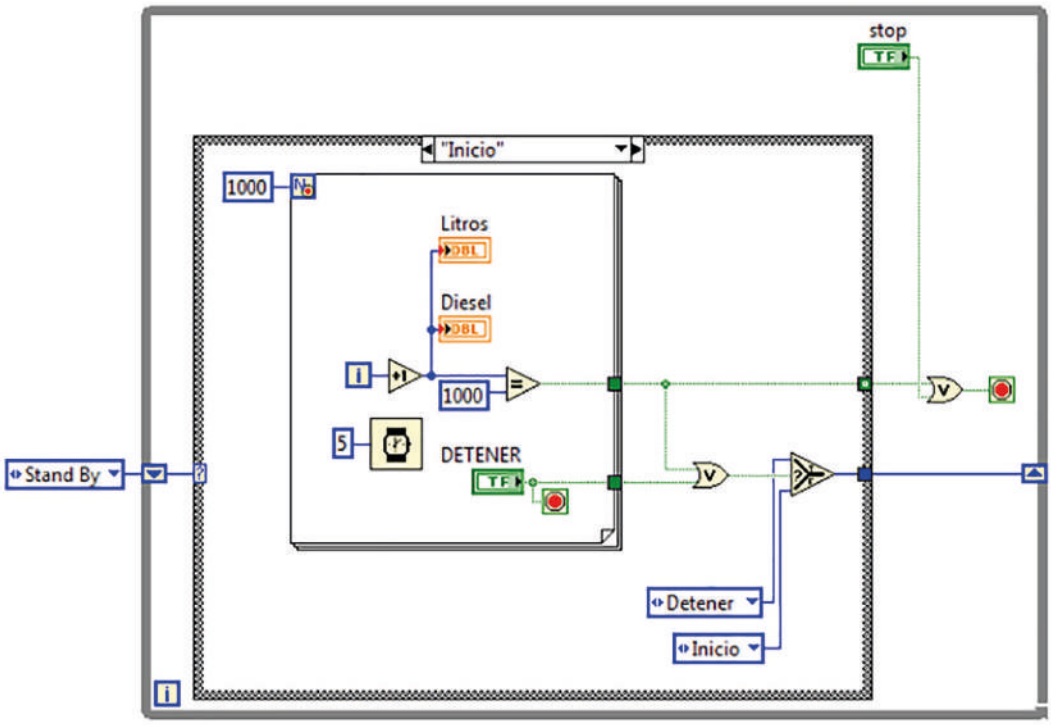



Figura 92. Diagrama de bloques del ejemplo de máquina de estados. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como apreciamos en la figura 92, al estar dentro del case Inicio, el llenado funciona gracias a un ciclo for, muy similar a ejercicios anteriores, el timing ayuda a elegir el tiempo total que tardará en llenarse.

Con un incremento en los indicadores, aumentando de uno en uno cada 5 milisegundos, al ser el valor de los indicadores en 1000, se detendrá el proceso de llenado.

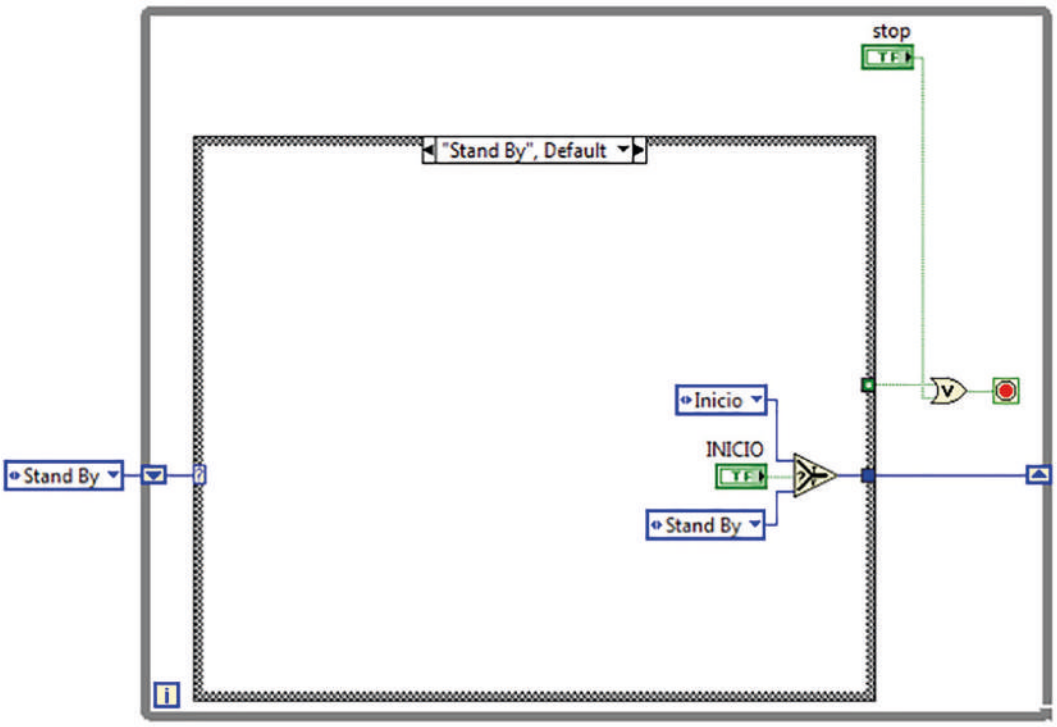



Figura 93. Caso Stand by. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al estar en modo de espera, el programa estará esperando a que el usuario presiona el botón de inicio para pasar al estado de llenar, si no, se quedará en su mismo estado

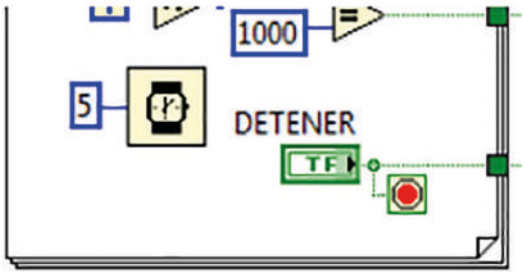



Figura 94. Funcionamiento del botón detener.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Durante el proceso de llenado, se puede parar el proceso de llenado del tanque sin detener la ejecución completa del programa, colocando el botón al for evitaremos que se pare completamente el programa.

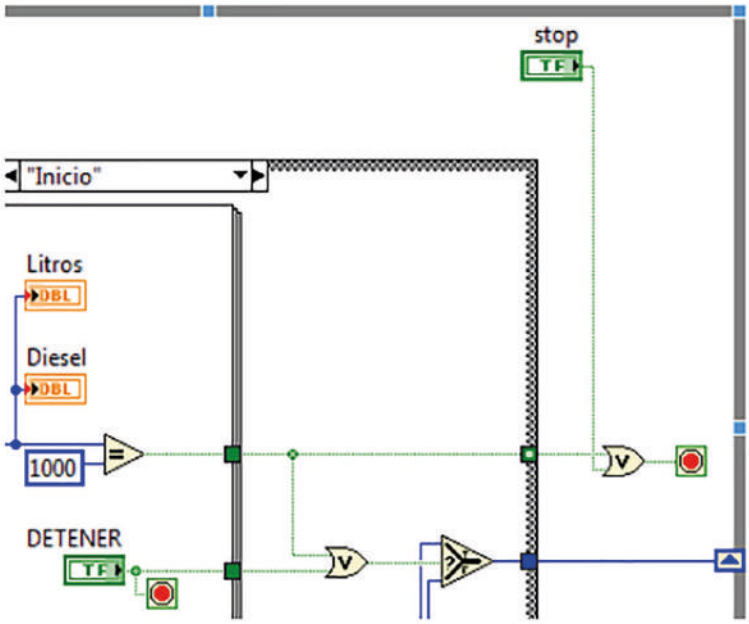



Figura 95. Paro total del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para detener por completo un programa, debemos usar la terminal del ciclo while, aquí se usará la compuerta lógica OR para que el programa se detenga si el tanque llega a mil litros o si el usuario presiona el botón de Stop.

Con esto daríamos por concluido el programa, queda solo arreglar algunos detalles visuales y estética que dependerá del programador.

## 2.7 Ejercicios Finales

Llegados a esta unidad, contamos con los conocimientos básicos para aprender a programar en LABVIEW, ya dependerá del estudiante comprender el funcionamiento de cada elemento y así resolver problemas relacionados

Veremos a continuación, unos ejemplos que ayuden a reforzar los conocimientos adquiridos.

Podemos iniciar con un ejercicio de alegra lineal, suponiendo que necesitamos resolver una multiplicación de matrices A X V, de esta forma podemos lograr que tanto Excel como Matlab

Teniendo en cuenta el algebra lineal podemos plantear un ejercicio donde se aplique el tema, por ejemplo, si necesitamos resolver una matriz A x B, cualquier software nos será de ayuda, especialmente LabVIEW.

Dentro de LabVIEW podremos crear un programa que se encargue de matricular matrices usando Arrays y todas las demás operaciones, pero sería un poco pérdida de tiempo hacer un programa que realice estas operaciones puesto que, las categorías ya contemplan ejemplos para entender el funcionamiento.

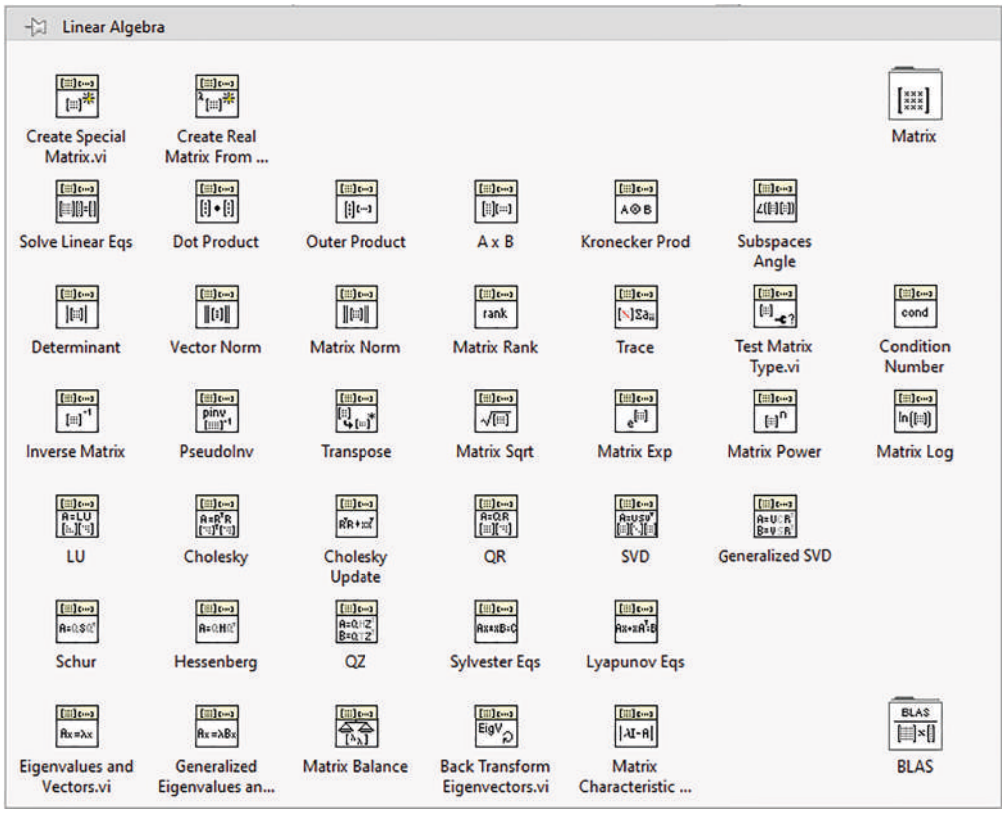



Figura 96. Menú de elementos para álgebra lineal. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para ingresar en esta categoría, seleccionamos en el menú de bloque la categoría Mathematics > Linear Algebra.

En el panel frontal, vamos a seleccionar tres matrices, la matriz A, B y C, siendo A y B, aquellas matrices que van a ser multiplicadas y C, la matriz resultada, con esto podemos saber que A y B serán controles y C la matriz indicadora.

Las matrices están disponibles en el menú, en la sección de Arrays, usaremos Real Matrix para matrices que contienen números reales, también podríamos usarla para números complejos seleccionando Complex Matrix.

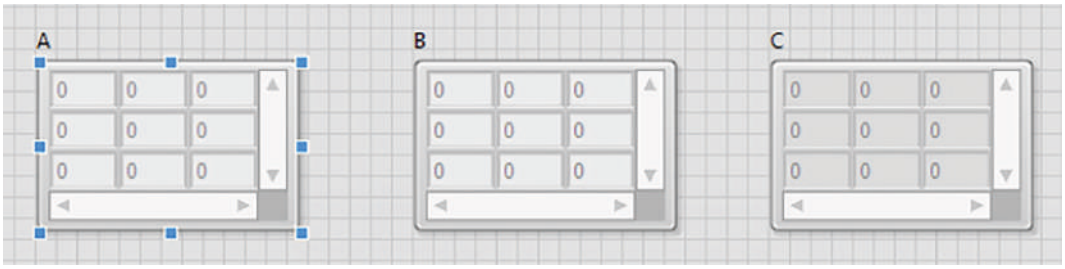



Figura 97. Matrices para el ejemplo de multiplicación de A X B. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Dentro del menú de algebra lineal encontramos el elemento A x B, el cual usaremos para conectar la matriz de la siguiente forma

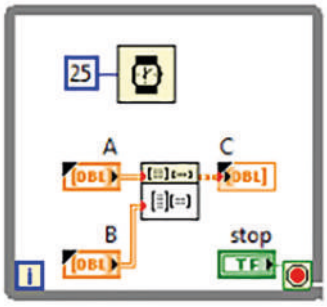



Figura 98. Diagrama de bloques del programa A X B.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La matriz A y B, la conectamos como controles, es ciclo while es opcional, pero se recomienda ya que mantendrá en ejecución el programa y actualizará los valores en tiempo real.

Debemos tener en cuenta que el orden de conexión de las matrices cambiara en relación al orden de posicionamiento de cada matriz.

Podemos trabajar matrices de distintos tamaños, para el ejemplo es de 3 x 3 pero podemos trabajar con matrices de N x N

## 2.8 Tamaño De Un Botón Con Nodos De Propiedad

Para este ejercicio, usaremos los nodos de propiedad que nos ayudarán a ajustar el tamaño de un botón, con el modo size, ajustamos los parámetros de Height y Width

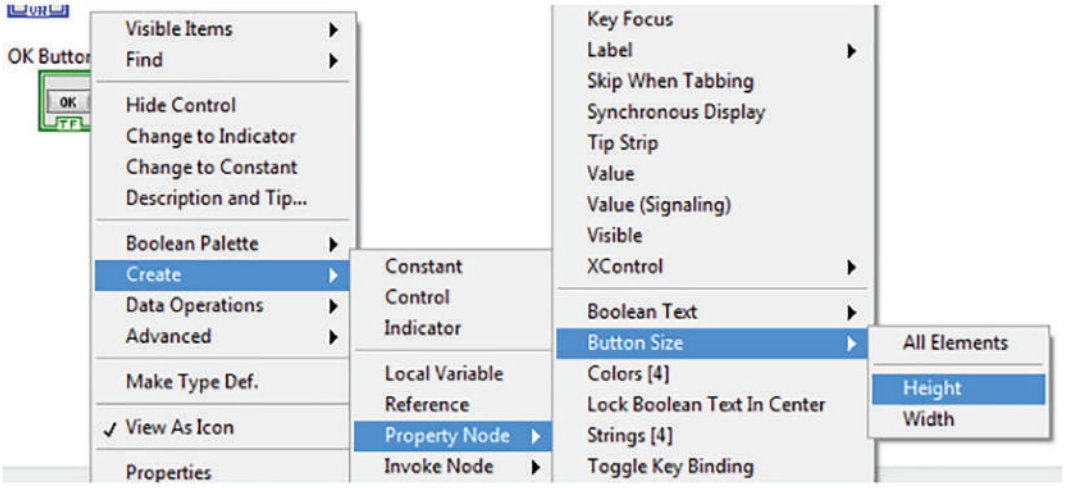



Figura 99. Ubicación del nodo de propiedad Size. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Con el nodo en cuestión, ajustamos el ancho y largo de un botón, para esto requerimos de dos controles tipo Slide y una conexión al Cluster del nodo de propiedad, mediante un Bundle by Name.

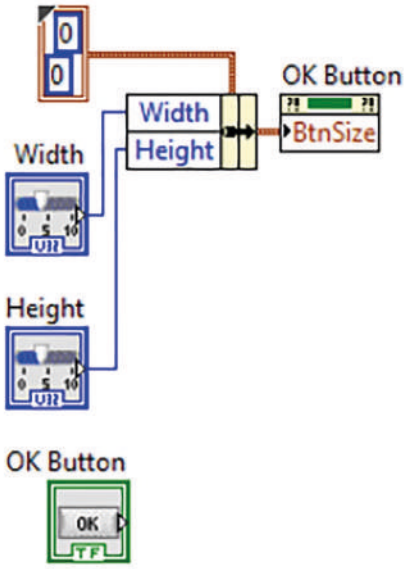



Figura 100. Código del programa con nodo de propiedad Size. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*


Figura 101. Panel frontal del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

## 2.9 Ejercicio Con Array De Leds

Dentro de este ejercicio, buscaremos abordar los conocimientos que se tiene sobre los Arrays, el ejercicio contempla una matriz de 11 x 11 y el objetivo de la practica es encender el led del centro e ir encendiendo los demás a su alrededor, como si de una onda o espiral se tratase.

Brindando la opción para que el usuario, elija la dirección en donde se encenderá el siguiente led, sea arriba, abajo, a la izquierda o a la derecha.

El indicador de texto nos ayudará a mostrar el mensaje para cuando el programa se ejecuta o se detiene, junto a un botón stop que dará salida al cierre del programa.

Tendremos el siguiente panel:


Figura 102. Panel frontal del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Y la programación de bloques es la siguiente:

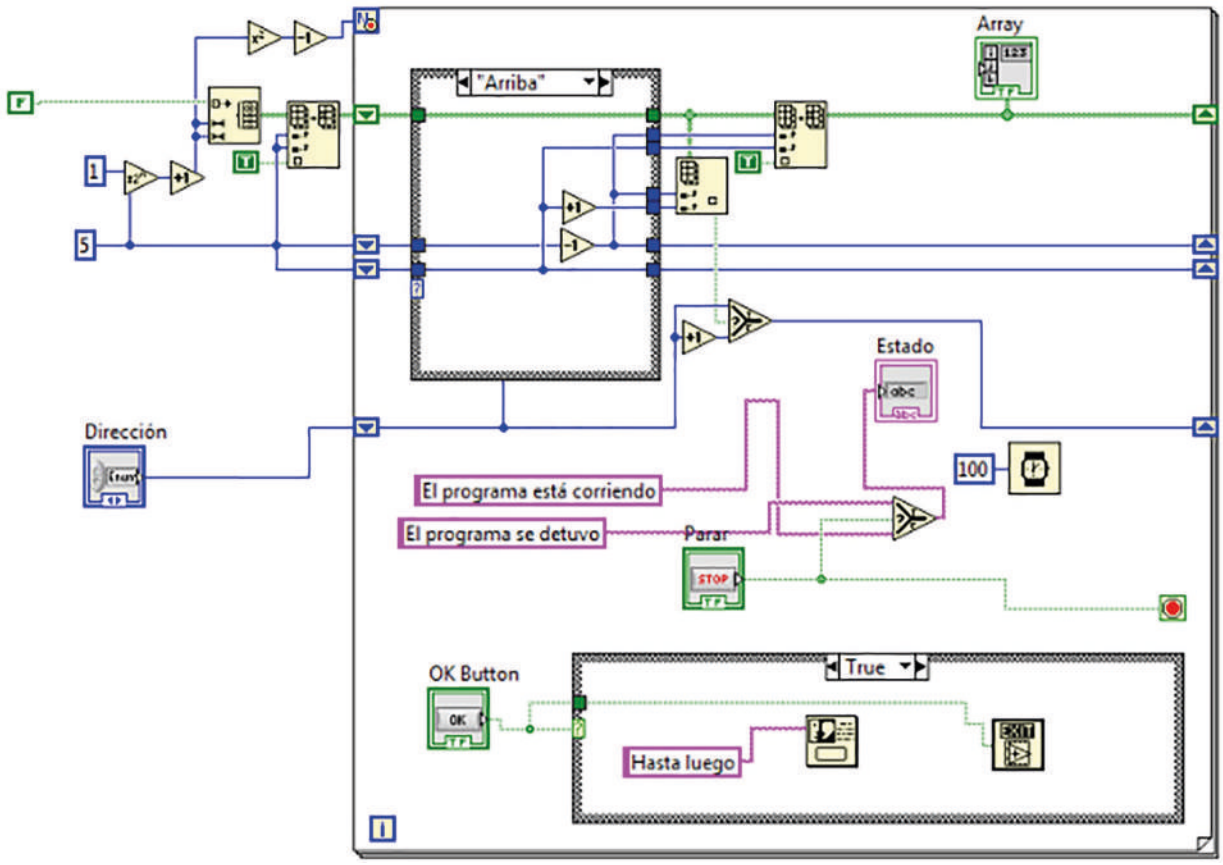



Figura 103. Diagrama de bloques del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para este programa, se implementa un ciclo for y dos estructuras case, la complejidad de este ejercicio se basa en determinar la dirección de encendido, el código será segmentado para una mejor explicación.

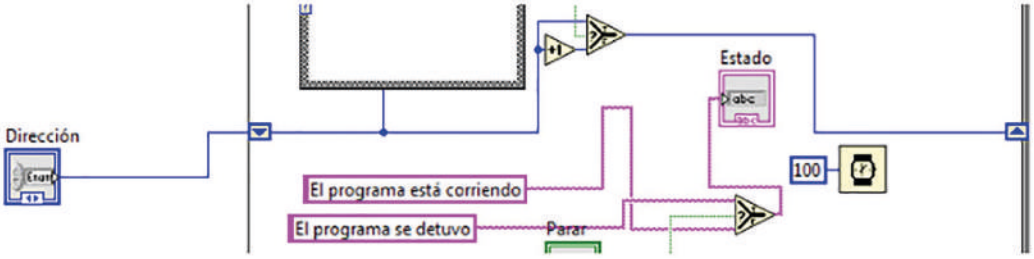



Figura 104. Uso de shift Register con enum. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Se emplea un Enum compuesto por cuatro opciones: arriba, abajo, izquierda y derecha. Este Enum es responsable de establecer la dirección en la que se activarán los LEDs. Se conecta a un registro de desplazamiento, que a su vez se vincula con una estructura case. Cada uno de los cuatro casos en dicha estructura tiene una configuración distinta, aunque difieren principalmente en el orden en que se producen los incrementos y decrementos.

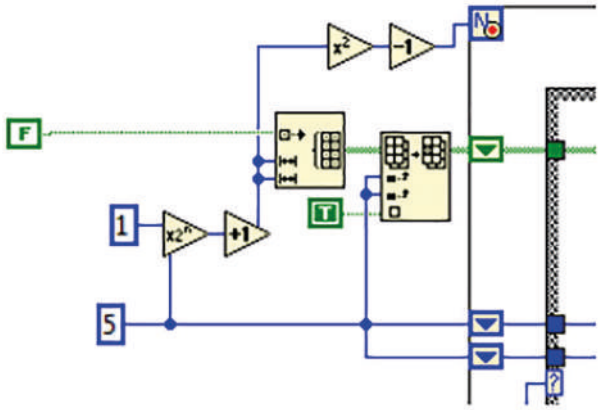



Figura 105. Configuración del ciclo for y del array de leds. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Se emplea un arreglo de inicialización para establecer la configuración inicial del array. La constante "False" indica que, al iniciar el programa, los LEDs estarán apagados. Las dos últimas terminales hacen referencia a la dimensión del arreglo, el cual es de 11 x 11. Para lograr esto, se usaron dos constantes, "5" y "1", donde "x" tiene un valor de 5 y "n" tiene un valor de 1. Al multiplicar "x" por 2 elevado a la potencia de "n", obtenemos un valor de "10", al cual se le añade un incremento para alcanzar un valor de "11". De esta manera, se establece que el arreglo es de 11 x 11. Todo lo anterior se puede sustituir por un simple 11 en cada terminal, para hacer las cosas más sencillas.

Luego podemos observar que el valor 11 se conecta a un elemento que eleva a la segunda potencia, lo que hace que nuestra X tenga un nuevo valor de 121, luego se conecta a un decremento dando como resultado 120.

El numero 120 representa el número de leds que, a su vez, representa la totalidad de repeticiones que se ejecutará

Nuestro resultado del array, se conectará a un subset, que será el encargado de reemplazar el estado del led, pasando de apagado a encendido con ayuda del shift Register.

El ultimo elemento se conecta al shift Register que pasa por la estructura case, luego se conecta al otro. Este subset seguirá el orden de encendido de los casos y dejará en un valor verdadero o positivo, los leds que se van encendiendo, para al final mostrar el resultado.

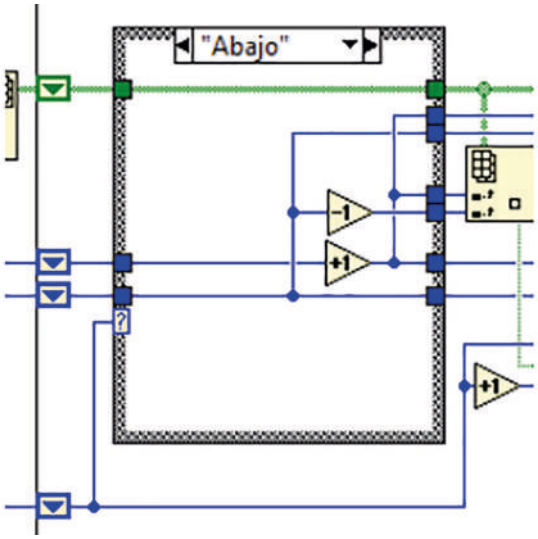



Figura 106. Estructura case para el orden de encendido. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para establecer la dirección de los leds, se usan dos elementos, para subir o bajar, con un decremento o aumento, se enciende el led, a la derecha o izquierda, dependiendo de el orden que se desea dar.

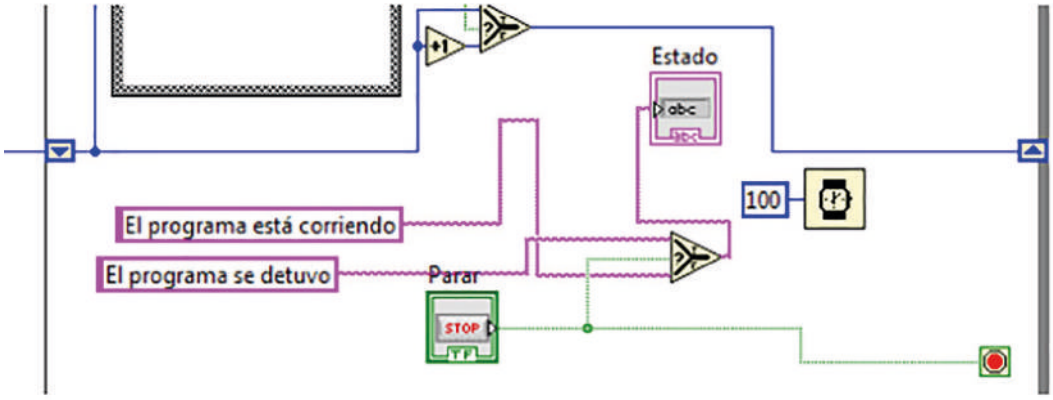



Figura 107. Textos para el estado del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para determinar el estado del programa, se muestra en el indicador de texto si el programa esta en ejecución o no, para esto simplemente se toman dos constantes de texto conectadas al Select y a su vez, al botón Stop, si el botón stop no se presiona, por defecto el mensaje que se mostrara es que el programa se encuentra ejecutando, al presionar el botón de stop, se mostrará un mensaje indicando que el programa se detuvo.

Se debe destacar que el timer que se ve en la figura 107 sirve para apreciar la visualización del encendido de los leds.

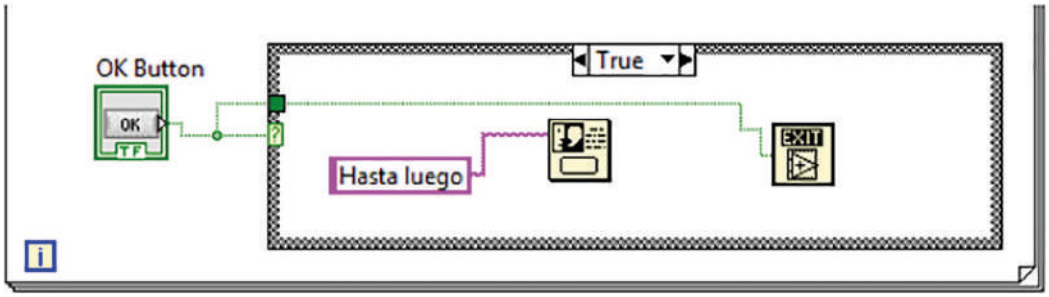



Figura 108. Mensaje de despedida y salida de LabVIEW. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para parar por completo nuestro programa, se conectará al case, cuando el botón de encendido cambie a true, pondremos una ventana emergente con la frase “hasta luego” posteriormente con elemento Exit, se saldrá de LabVIEW.

## 2.10 Máquina Tragamonedas

Esta práctica tiene un mayor grado de complejidad, es extenso y se utilizan muchos elementos y varios ciclos, simula una maquina de tragamonedas, como de un casino o de esas que podíamos encontrar en las tiendas hace unos años.

El funcionamiento es básico, se ingresa una moneda, se escoge una fruta, cada fruta tiene un valor distinto, valor el cual el usuario ganará si se enciende aquel led que el usuario selecciono, al insertar el dinero para apostar y presionar el botón de la fruta, comenzará a girar a alta velocidad y posteriormente empezará a disminuir su velocidad, es una simulación en la que se enciende y se apaga un led, presentando una animación

Ganar es sencillo, solo debe encender el led de la fruta que seleccionamos, se gana el valor que contenga la fruta y en caso de perder, se descuenta el valor apostado de nuestro dinero total.

Cuando se presione el botón de jugar sin realiza una apuesta o ingresar un mensaje, se debe mostrar un mensaje de que no hay dinero suficiente para apostar, si el usuario gana, se debe mostrar también un mensaje donde felicite al jugador por ganar.

Para retirar el dinero, se debe crear un botón que, al presionarlo, comience a disminuir del dinero total, de forma periódica, simulando que las monedas van saliendo de la máquina.

El panel frontal quedará un poco extenso y se utilizarán imágenes reales para lograr una mejor simulación


*Figura 109. Panel frontal del programa "Máquina tragamonedas". Autoría Propia.*

El panel frontal estará compuesto en su mayoría por leds, cada uno para cada fruta, los botones de la derecha y el control numérico, el indicador y el botón de parar.

El código será explicado por partes debido a su magnitud:

Inicialmente, el ciclo While mantendrá en ejecución el programa hasta que se presione el botón Stop, simulando el encendido o apagado de la máquina.

Dentro de este ciclo while, se colocará el botón de jugar, que estará conectado a un case, si el caso es falso no se hace nada, al ser verdadero se debe evaluar si el control numérico es mayor que 0, con esto buscamos que nuestro programa valide si hay dinero disponible para apostar.

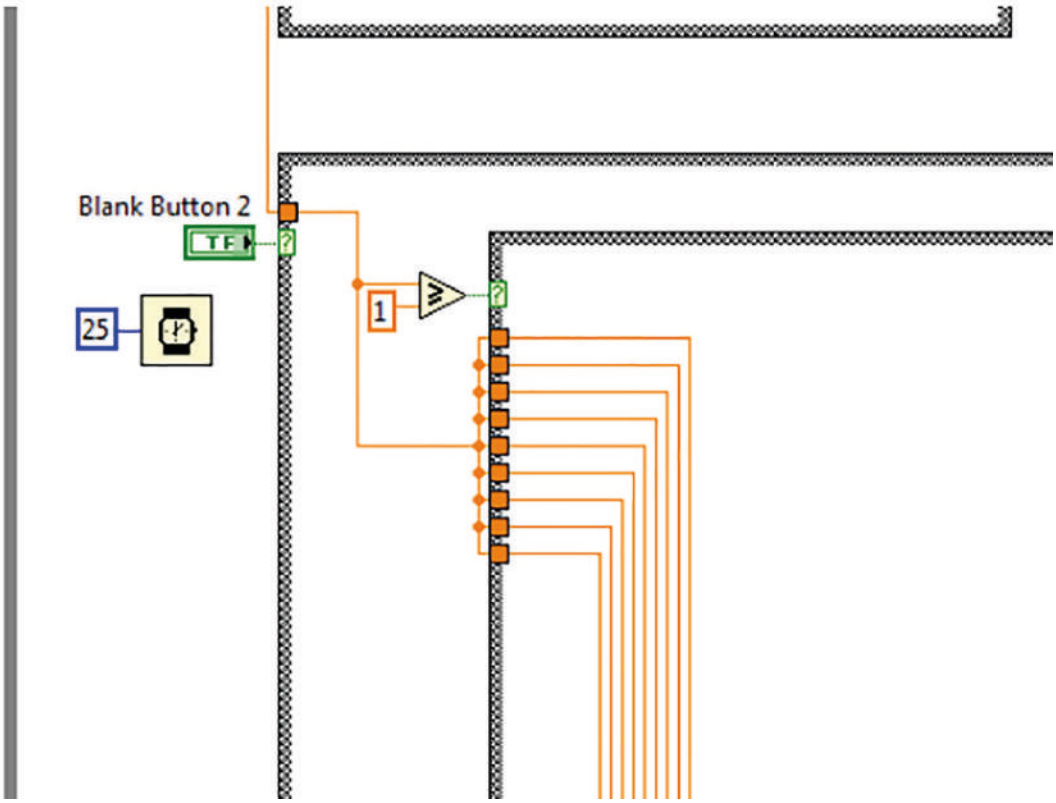



Figura 110. Se evalúa si dinero es mayor o igual a 1. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Lo mas interesante de este programa, radica en la simulación del recorrido del led por todas las frutas encendiendo y apagando de forma sincronizada y disminuyendo la velocidad de apagado.

Será útil realizar un ciclo for, sin embargo, como el programa se basa en un juego de azar, no sería correcto definir la cantidad de vueltas o cambios que hará el led, esto debe ser totalmente aleatorio.

Podemos usar un numero random o aleatorio, este estará disponible en el menú de diagramas numéricos, teniendo el bloque, definimos un rango para el valor, estando entre 0 y 150.

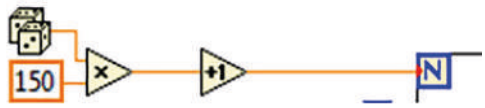



Figura 111. Configuración del número de repeticiones. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ahora, para simular la velocidad y disminución del led, usaremos un contador con shift Register, el cual se inicializa en cero y se conecta a un incremento y este a su vez, a un timer.

Esto hará que el timer comience en cero y vaya avanzando conforme a las repeticiones del ciclo.


Figura 112. Configuración de la velocidad de avance del led. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para que durante el ciclo for, los leds no vayan quedando encendidos, iniciamos el Register en true y los demás que corresponden a cada led, se inician en false, esto hará que los leds no se queden encendidos y solo se encenderá uno a la vez.

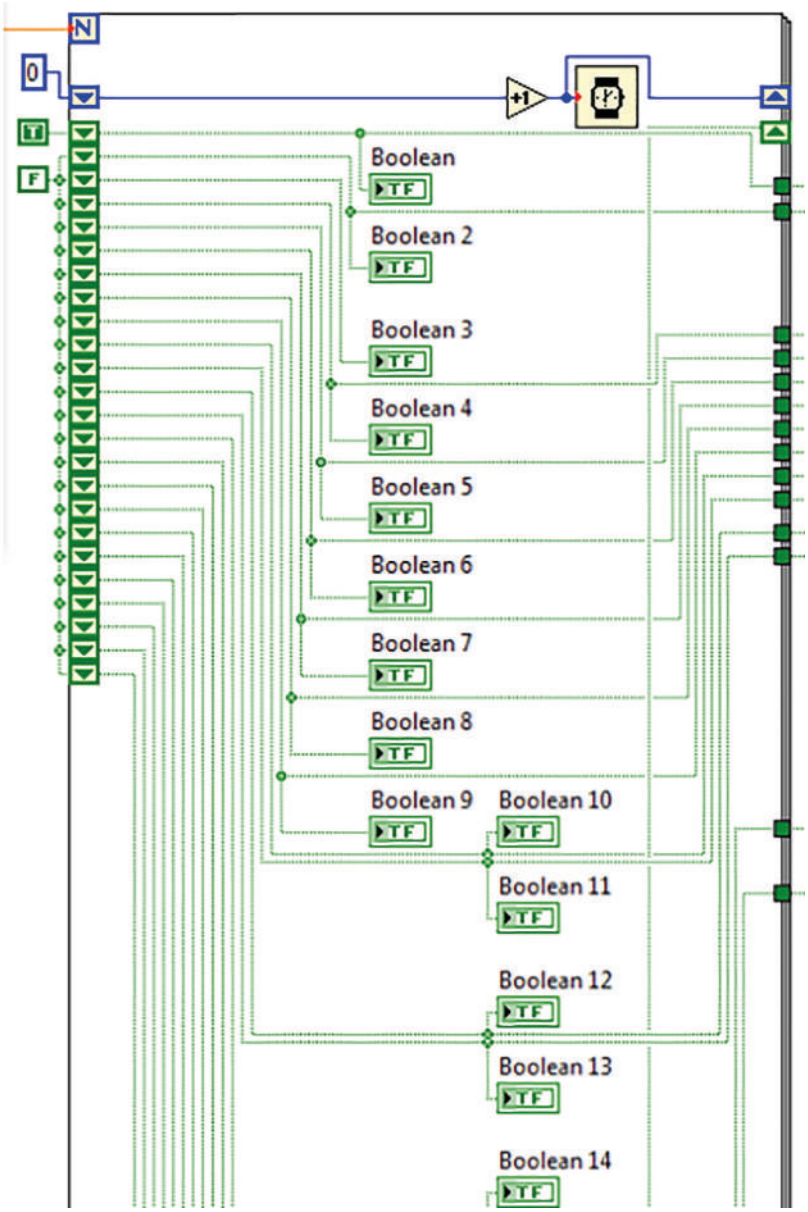



Figura 113. Inicialización de shift registers. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al ejecutar el programa, el indicador de texto nos muestra el mensaje, dicho mensaje se encuentra dentro del while y se mostrará durante la ejecución

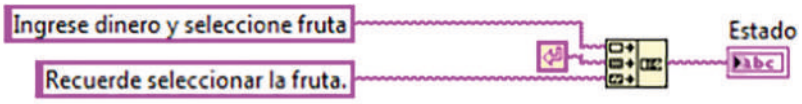



Figura 114. Mensaje de texto del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Se utilizarán dos frases para interactuar con el usuario, por lo que se requiere de un Carriage Return Constant, que ayudará a colocar una oración debajo de otra

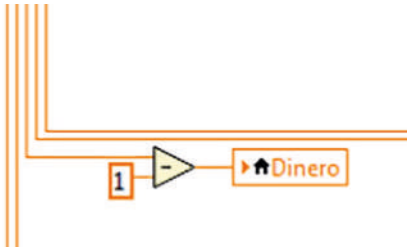



Figura 115. Resta del dinero del botón “jugar”. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Conectamos ahora, el control numérico a la operación de la resta, el resultado estará enlazado a la variable local del mismo control

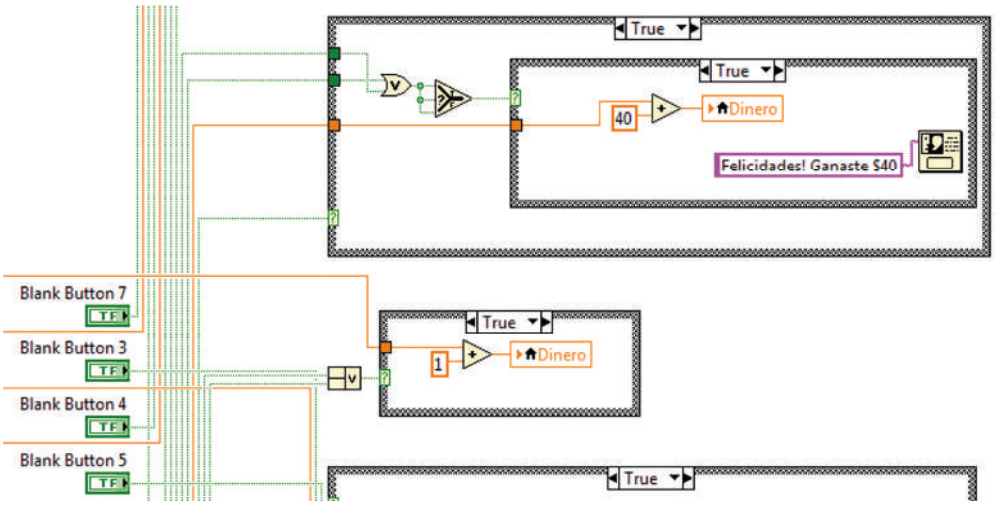



Figura 116. Conexión de los botones. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Cada botón estará representado por una fruta, uniéndolos a la estructura del caso, se utiliza una compuerta lógica OR, donde se pretende que cada X led, se queda encendido entonces se cumplirá el caso, habilitando las demás opciones.

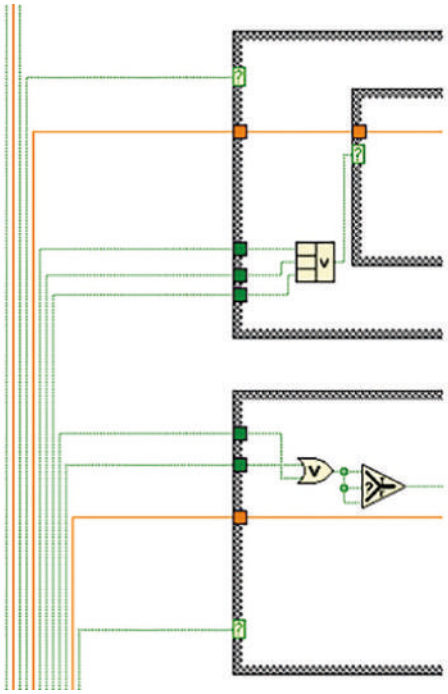


Figura 117. Uso de compuerta AND. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La compuerta estará conectada a otro case, donde el caso false esta vacío pero el verdadero si tiene operación por realizar

Por ejemplo, en el caso de la naranja, la estructura case es la siguiente:

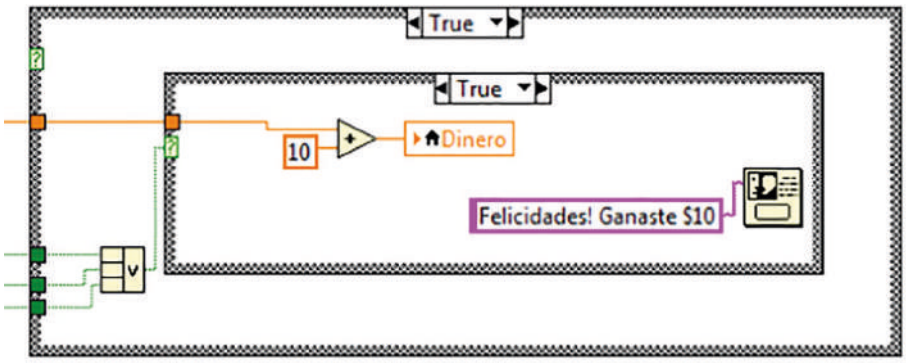



Figura 118. Casos para leds representativos de la naranja. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La fruta naranja aparece tres veces en el tablero, lo que significa que se utilizan tres LEDs para representarla. Si uno de estos LEDs permanece encendido al finalizar el ciclo "for", se utiliza una operación de suma para añadir el valor de la fruta (en este caso, 10) al dinero del usuario. Para llevar a cabo esta tarea, se utiliza una variable local denominada "dinero".

También se utiliza el elemento One Button Dialog para mostrar un mensaje emergente

Se utiliza igualmente para cada botón, entonces se tendrán nueve estructuras.

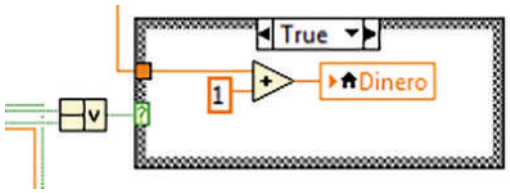



Figura 119. Casos para salto de turno. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Si el led se detiene en la figura del muñequito que hay a los lados del tablero, sólo se regresa la cantidad que el usuario gastó.

Para terminar, nos queda la parte donde el usuario decide cobrar el dinero que ganó o el dinero que ingresó.

Para esto utilizaremos otro contador, pero esta vez descendente.

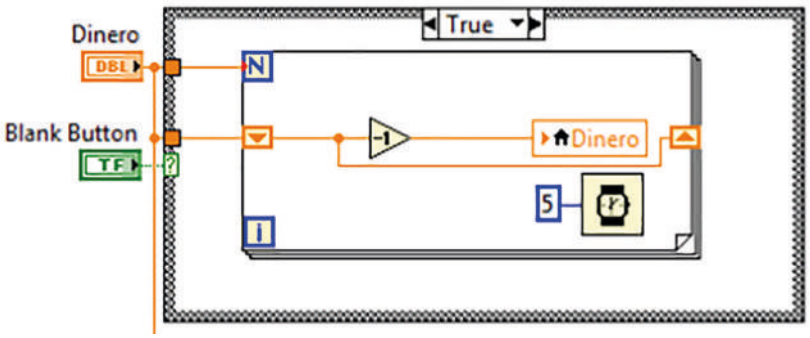



Figura 120. Contador descendente. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El botón de cobrar, estará conectado a la estructura case, dicho caso falso no hará nada, pero el verdadero utilizará el ciclo for.

Las iteraciones del ciclo for estará dado por la cantidad que haya en el control numérico, el shift Register inicializará el control numérico, que estará conectado a un decremento y a su vez, a una variable local del mismo control.

Con el timer de 5 ms, se determinará la velocidad del contador, a su vez, al presionar el botón, los números del control irán descendiendo simulando así, la caída del dinero en la máquina.

Se debe dejar claro que, la configuración para el botón de cobrar, quedará bajo una acción mecánica del Switch When Pressed, con esto se busca que, mientras se mantenga presionado el botón, se ejecute la acción, si se presiona y posteriormente se suelta, se ejecutará el ciclo una o dos veces, al mantenerlo presionado se deberá ejecutar hasta quedar todo en 0, simulando un realismo superior.

Esto podemos aplicarlo a las frutas y sus respectivos botones.

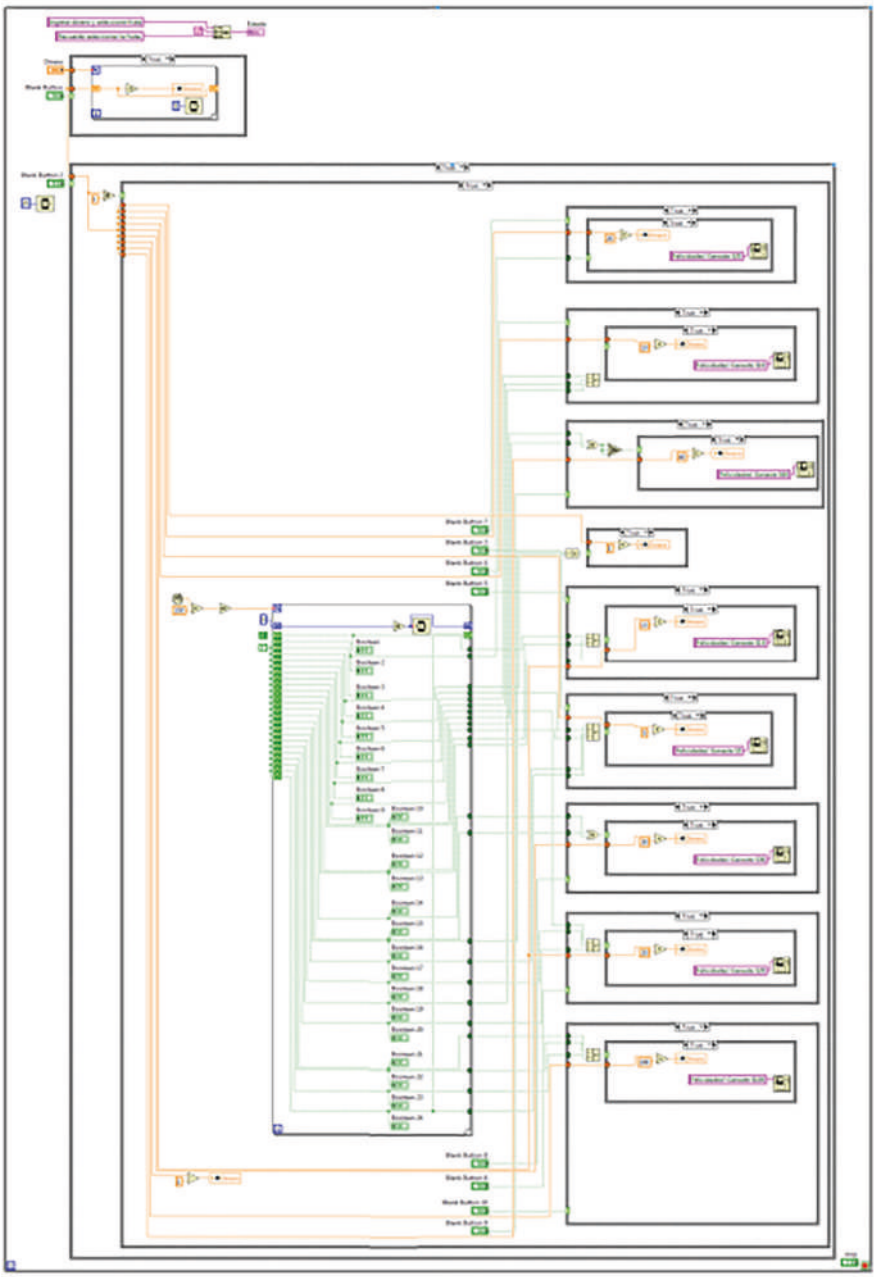


Figura 121. Código completo de la máquina tragamonedas.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

## 2.11 Edición Del Ícono De Un Programa

Para editar el icono de un programa que ya hemos creado, sencillamente iremos a la esquina derecha superior y daremos clic sobe el icono, posteriormente en Edit Icon.

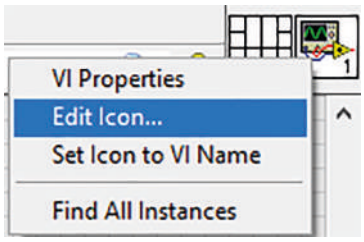



Figura 122. editor de íconos. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

También se puede acceder dando doble clic al ícono.

Al hacer eso, nos aparecerá una ventana como la siguiente:

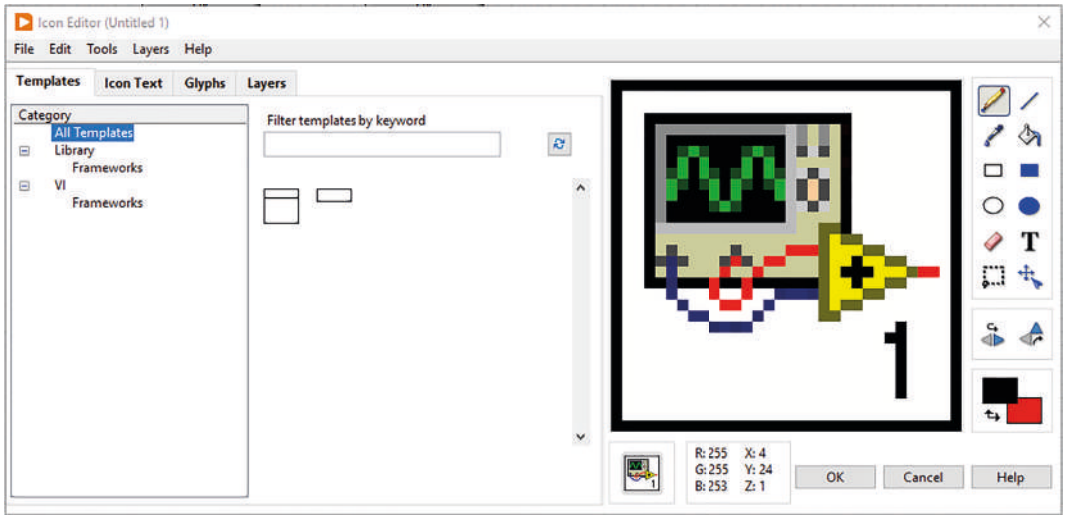



Figura 123. Editor del ícono del programa. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ahí podemos editar el ícono como se requiera, se pueden utilizar las imágenes de la pestaña Glyphs.

Y desde ahí se puede editar el ícono de la forma que se requiera, usando imágenes, las herramientas del editor, entre otros.

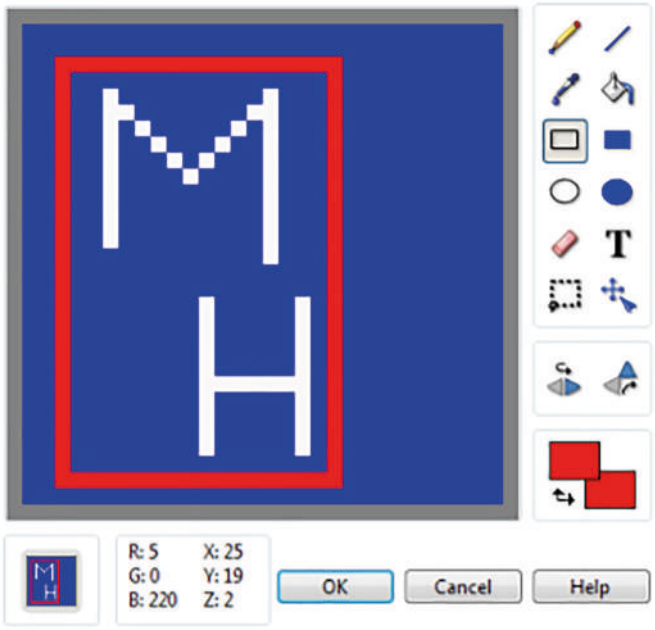



Figura 124. Ejemplo de ícono. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

# CAPÍTULO 3

## 3.1 Aplicaciones Prácticas Con Sistemas Embebidos

La medición y la automatización basadas en computadora son populares porque nuestra PC proporciona la plataforma necesitamos hacer que nuestros sistemas de medición y automatización sean confiables y eficientes. Su extensa Las capacidades de procesamiento nos permiten crear soluciones flexibles basadas en los estándares de la industria. Con esta flexibilidad, podemos ajustar las especificaciones de nuestra aplicación más fácilmente que con las herramientas tradicionales.

El software de National Instruments, que incluye LabVIEW y Measurement Studio, ofrece análisis de datos, conectividad y poder de presentación basados en PC a nuevos niveles en medición y aplicaciones de automatización. El hardware y el software de National Instruments conectan la computadora a su aplicación. Al proporcionar una amplia selección de hardware, incluida la adquisición de datos y dispositivos de acondicionamiento de señales, interfaces de control de instrumentos (como GPIB, Serial, VXI y PXI), adquisición de imágenes, control de movimiento e interfaces de comunicaciones industriales, National Instruments ofrece la más amplia gama de soluciones para prácticamente cualquier medida como se muestra en la Figura 10.1.

Este capítulo describe el control de instrumentos de instrumentos autónomos utilizando un GPIB o serial. interfaz. Use LabVIEW para controlar y adquirir datos de instrumentos con Instrument I/O Assistant, VISA API y controladores de instrumentos. Si elige el control estándar de la industria tecnologías, no está limitado al tipo de instrumento que puede controlar. Puedes mezclar y combinar instrumentos de varias categorías como se muestra en la Figura 10.2. Las categorías más comunes de Las interfaces de instrumentos son GPIB, serie, instrumentos modulares e instrumentos modulares PXI. Adicional Los tipos de instrumentos incluyen adquisición de imágenes, control de movimiento, USB, Ethernet, puerto paralelo, NI-CAN y otros dispositivos.

## 3.2 Formas De Comunicación Entre LabVIEW Y Sistemas Embebidos

Para poder realizar desarrollos de sistemas y adquisición de datos por medio de sistemas embebidos, se debe entablar una comunicación entre el software LabVIEW y el embebido, para este caso los ejercicios que desarrollaremos será bajo la comunicación serial.

No solo basta entablar el protocolo de comunicación, si no la forma de cómo se gestionarán los datos, una forma podría ser que el sistema embebido se encargue de la gestión de los datos y los envíe a LabVIEW por medio del protocolo de comunicación serial. Otra forma sería por medio de firmware y librerías propias asociado al sistema embebido utilizado, para casos comunes y por facilidad se utiliza por ejemplo la librería LINX de MakerHub.

Realizando la gestión directa por el sistema embebido no habría que utilizar Toolbox adicionales, simplemente nos apropiamos de las funciones del VISA para poder entablar la comunicación serial, ver figura.

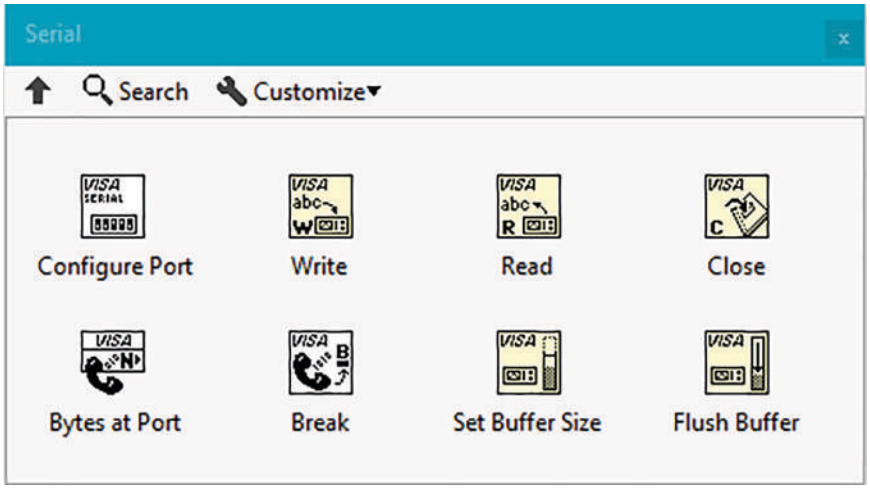



Figura 125. Librería de comunicación Serial VISA. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para las librerías como Linx, se cuenta con las siguientes Toolbox:

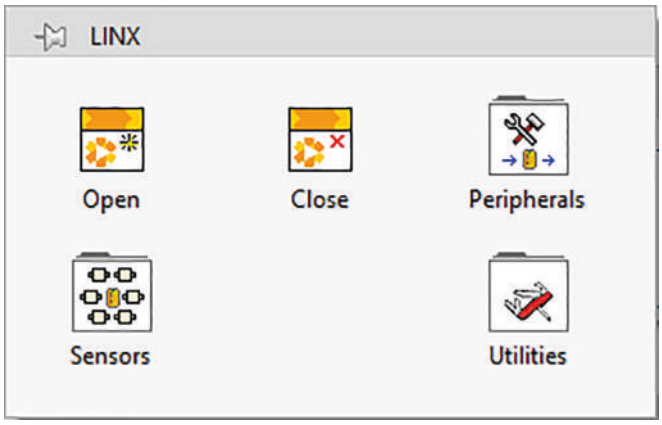



Figura 126. Librería LINX. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

A partir de este momento nos adentramos en una serie de prácticas en donde tomamos como referencia el sistema embebido Arduino, para poder desarrollas las aplicaciones planteadas en cada una de las guías.

### 3.2.1 **Practica #1. Envío De Datos De Arduino – LabVIEW, Con Gestión Desde La Arduino**

Esta práctica estará orientada en el funcionamiento de NI-VISA, la comunicación y la trasmisión de datos con el sistema embebido Arduino enfocando el entorno hacia a la instrumentación virtual ofrecida por LabVIEW. La figura 127 muestra un esquema sencillo para comenzar con la práctica.

Posteriormente, se realiza el diseño de simulación en Tinkercad, cuyo diagrama esquemático consta de la siguiente arquitectura, véase la figura. El diseño realizado permite el envío de datos por el puerto COM, cada vez que se presiona el pulsador, que se conectará con el instrumento virtual diseñado en LabVIEW.

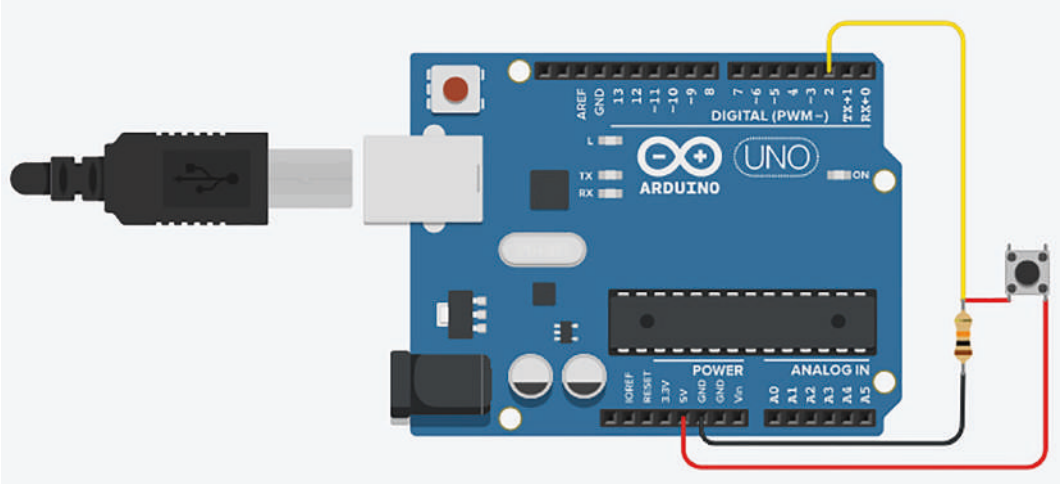



Figura 127. Esquemático para aplicación práctica 1 Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

Continuaremos con el desarrollo de la práctica, evidenciando como sería el código de programación bajo el lenguaje de C, lenguaje utilizado por Arduino para el desarrollo de la programación. (Integrated Development Environment, IDE) de Arduino. El código de la aplicación, véase la figura, permite enviar un dato prestablecido si se presiona el pulsador, por el puerto virtual COM y lo envía a LabVIEW mediante un COM. En caso contrario termina enviando otro dato diferente.

```
void setup() {
 Serial.begin(9600);
 pinMode(2,INPUT);
}

void loop() {
if(digitalRead(2)==HIGH){
  Serial.write("A");
  }else{
    Serial.write("B");
    }
}
```

Figura 128. Código de la aplicación práctica 1. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El software diseñado por LabVIEW, tomará inicialmente los datos recibidos por el puerto NI VISA y mostrará el label, dentro de la programación de diagramas, se deberá parametrizar el puerto de comunicación NI VISA

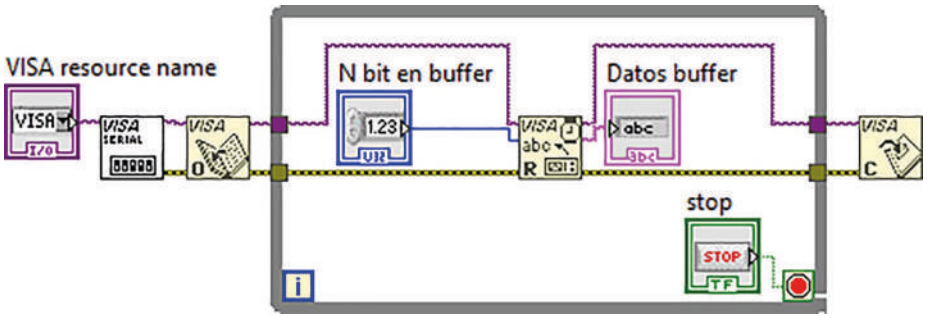



Figura 129. Programación de la aplicación práctica 1. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Durante la ejecución del programa se debe tener en cuenta que la configuración de los periféricos como el puerto, COM, que recibirá datos de un puerto serial y se hará la lectura de estos en LabVIEW.

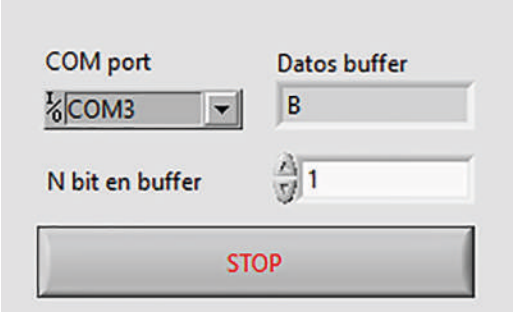



Figura 130. Panel de simulación del instrumento virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.2.2 **Practica #2. Recepción Y Envío De Datos De Arduino – LabVIEW**

El objetivo de esta práctica es afianzar los conocimientos en comunicación bidireccional utilizando el NI VISA de LabVIEW. En la figura 130 se muestra el diagrama esquemático con cada uno de los elementos del circuito electrónico que se usan.

Posteriormente, se realiza el diseño de simulación en Tinkercad, cuyo diagrama esquemático consta de la siguiente arquitectura, véase la figura. El diseño realizado permite el envío de datos desde el instrumento virtual diseñado en LabVIEW, luego el sistema embebido la recibe y nuevamente la retorna para que sea visualizada en la interfaz desarrollada en LabVIEW.

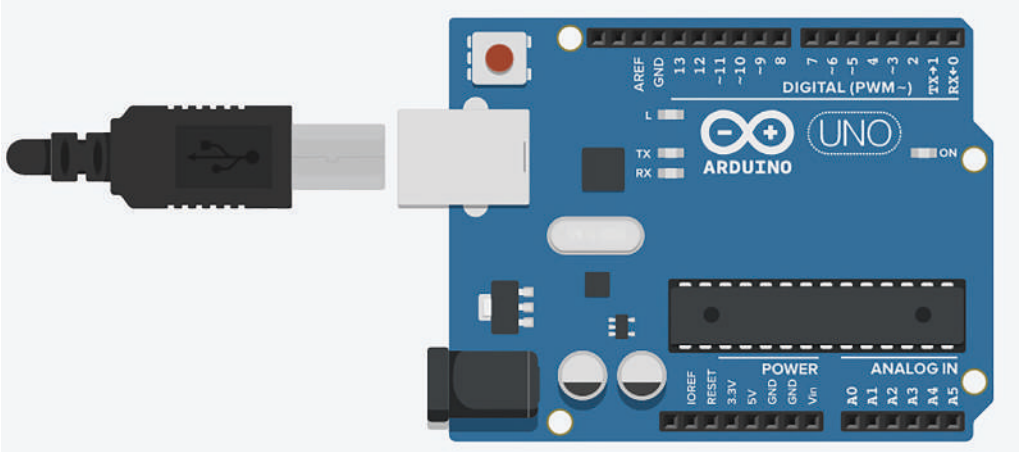



Figura 131. Esquemático para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

El código del sistema embebido, véase la figura, permite recibir un dato que llega desde la interfaz del LabVIEW, para luego reenviarlo a LabVIEW mediante un COM.

```
void setup() {
 Serial.begin(9600);
}

void loop() {
  if(Serial.available()){
  char inChar = Serial.read();
  Serial.println(inChar);
  delay(100);
  }
}
```

Figura 132. Código del envío de caracteres. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El software de instrumentación virtual LabVIEW diseñado envía los datos por el puerto NI VISA la Arduino lo recibe y luego reenvía el mismo dato al LabVIEW para luego poderlo visualizar en la interfaz de LabVIEW. En el diagrama de bloque deberá parametrizarse cual puerto se usará como comunicación de NI VISA y qué deberá hacer, una lectura de la cantidad de bits indicados.

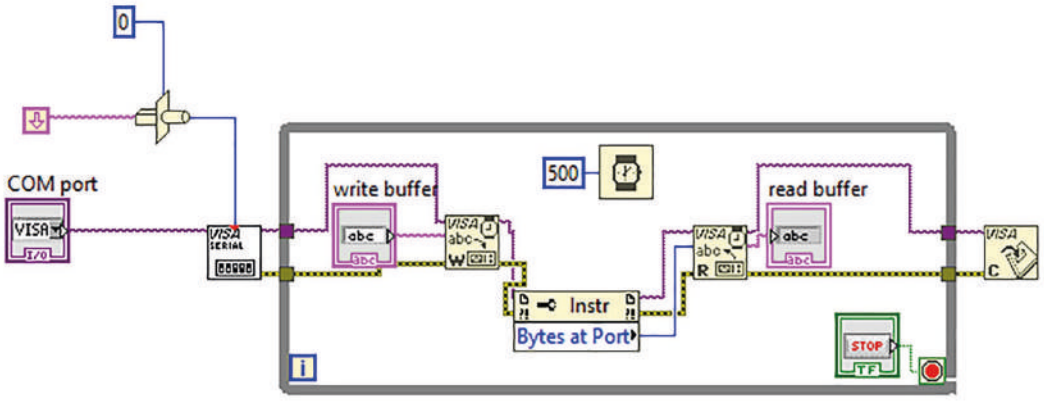



Figura 133. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Mientras se ejecuta el programa, es necesario tener en cuenta la configuración de los diferentes periféricos el COM, el cual recibirá datos desde el puerto serial y se hará la lectura de estos en LabVIEW.

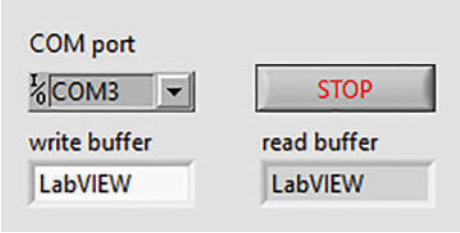



Figura 134. Panel de simulación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

## 3.3 Entradas Y Salidas Digitales

En estas prácticas nos vamos a centrar en lo más sencillo de la parte digital que es escribir y leer unos y ceros. Para la parte electrónica usaremos diodos led como salidas para mostrar un (on/off), también se usarán pulsadores los cuales proporcionan información digital de entrada.

### 3.3.1 Practica #3. Generar Salidas Digitales Con Gestión Desde La Arduino

Para esta práctica se controlará el encendido y apagado de diodos leds mediante pines digitales utilizando la herramienta NI-visa. Los elementos a necesitar para esta práctica son diodos led, resistencia y una tarjeta Arduino.

En la figura se muestra el diseño esquemático. El Arduino gestionará todos los datos digitales y a la vez se encuentra a la espera de recibir datos desde la interfaz diseñada en LabVIEW, la misma que servirá como visualizador de las asignaciones discretas en los pines digitales previamente configurados.

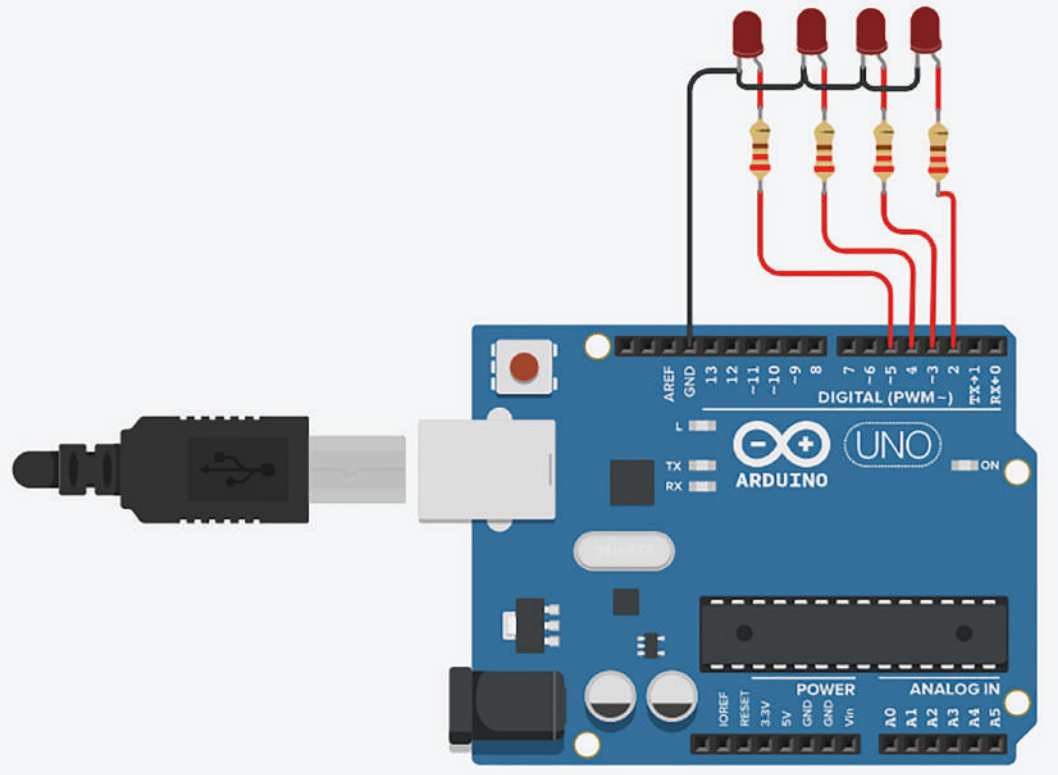



Figura 135. Esquemático de simulación en Isis Proteus para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

El código en la Arduino se encargará de obtener un dato enviado desde el LabVIEW y realiza una comparación frente a unas variables ya prestablecidas mediante una sentencia if y realizará la gestión de activación o inactivación de un puerto digital respectivamente en la Arduino.

Posteriormente, se asignará en el inicio la comunicación serial, dicha configuración de pines de salidas y entradas, inicializará variables en 0

Y, por último, se debe realizar una comparación cuando el puerto serial pueda leer un dato del buffer, por lo que se compara dicho valor con una serie de posibles casos para encender y apagar los LEDS.

```
char dato
int led1=2;int led1=3;int led1=4;int led1=5;
void setup() {
  Serial.begin(9600);
  pinMode(led1,OUTPUT);pinMode(led2,OUTPUT);pinMode(led3,OUTPUT);pinMode(led4,OUTPUT);
  digitalWrite(led1,LOW);digitalWrite(led2,LOW);digitalWrite(led3,LOW);digitalWrite(led4,LOW);
}
void loop() {
  while(Serial.available()){
    dato=(char)Serial.read();
    if(dato=='a'){digitalWrite(led4,LOW);digitalWrite(led3,LOW);digitalWrite(led2,LOW);digitalWrite(led1,LOW)))
    if(dato=='b'){digitalWrite(led4,LOW);digitalWrite(led3,LOW);digitalWrite(led2,LOW);digitalWrite(led1,HIGH)))
    if(dato=='c'){digitalWrite(led4,LOW);digitalWrite(led3,LOW);digitalWrite(led2,HIGH);digitalWrite(led1,LOW)))
    if(dato=='d'){digitalWrite(led4,LOW);digitalWrite(led3,LOW);digitalWrite(led2,HIGH);digitalWrite(led1,HIGH)))
    if(dato=='e'){digitalWrite(led4,LOW);digitalWrite(led3,HIGH);digitalWrite(led2,LOW);digitalWrite(led1,LOW)))
    if(dato=='f'){digitalWrite(led4,LOW);digitalWrite(led3,HIGH);digitalWrite(led2,LOW);digitalWrite(led1,HIGH)))
    if(dato=='g'){digitalWrite(led4,LOW);digitalWrite(led3,HIGH);digitalWrite(led2,HIGH);digitalWrite(led1,LOW)))
    if(dato=='h'){digitalWrite(led4,LOW);digitalWrite(led3,HIGH);digitalWrite(led2,HIGH);digitalWrite(led1,HIGH)))
    if(dato=='i'){digitalWrite(led4,HIGH);digitalWrite(led3,LOW);digitalWrite(led2,LOW);digitalWrite(led1,LOW)))
    if(dato=='j'){digitalWrite(led4,HIGH);digitalWrite(led3,LOW);digitalWrite(led2,LOW);digitalWrite(led1,HIGH)))
    if(dato=='k'){digitalWrite(led4,HIGH);digitalWrite(led3,LOW);digitalWrite(led2,HIGH);digitalWrite(led1,LOW)))
    if(dato=='l'){digitalWrite(led4,HIGH);digitalWrite(led3,LOW);digitalWrite(led2,HIGH);digitalWrite(led1,HIGH)))
    if(dato=='m'){digitalWrite(led4,HIGH);digitalWrite(led3,HIGH);digitalWrite(led2,LOW);digitalWrite(led1,LOW)))
    if(dato=='n'){digitalWrite(led4,HIGH);digitalWrite(led3,HIGH);digitalWrite(led2,LOW);digitalWrite(led1,HIGH)))
    if(dato=='o'){digitalWrite(led4,HIGH);digitalWrite(led3,HIGH);digitalWrite(led2,HIGH);digitalWrite(led1,LOW)))
    if(dato=='p'){digitalWrite(led4,HIGH);digitalWrite(led3,HIGH);digitalWrite(led2,HIGH);digitalWrite(led1,HIGH)))
    }
}
```

Figura 136. Código de programación de configuración de variables para aplicación. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la figura 137, se muestra el diseño de diagrama de bloques en LabVIEW, dicho código se encarga de pasar el dato de cada botón, aplicando un mapeo de cada estado, al tener 4 pulsaciones, se tendrá 16 posibles combinaciones, cada una enviará el dato al buffer, mediante el puerto COM.

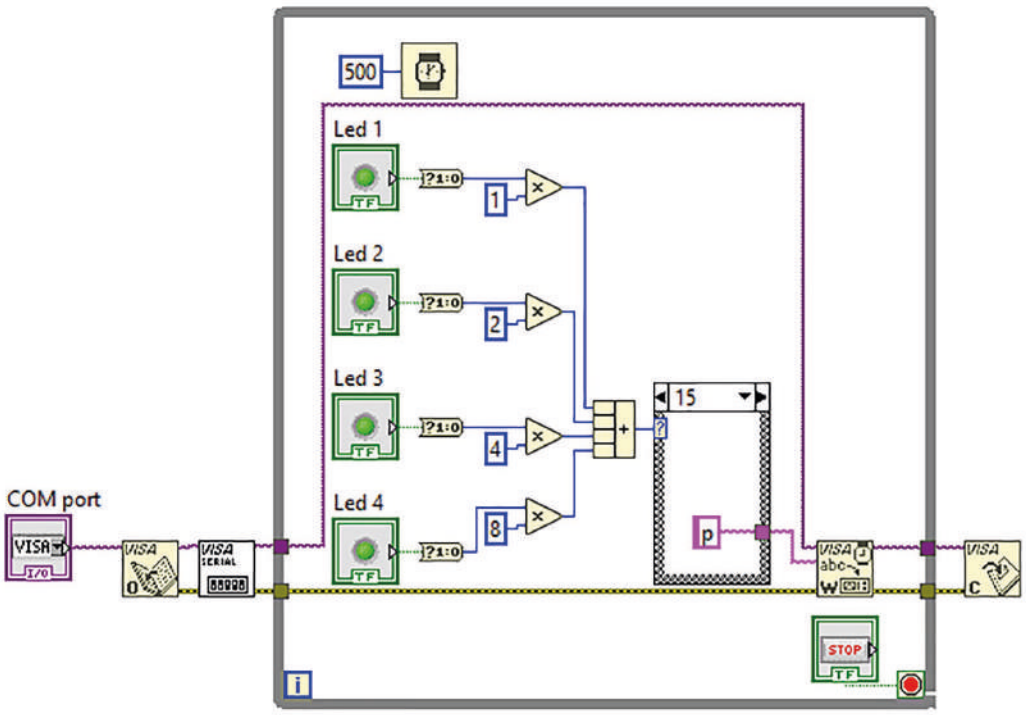



Figura 137. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al momento de la ejecución desde la interfaz de LabVIEW, se enviarán los valores de acuerdo a los estados de los botones y al número de combinaciones previamente programadas en la Arduino, tal como se mostró en la figura.

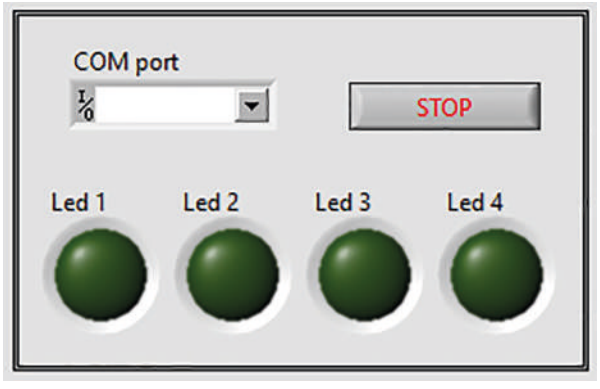



Figura 138. Panel frontal de instrumentación virtual de la aplicación.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.3.2 **Practica #4. Generar Salidas Digitales Con LINX**

Para esta práctica se controlará el encendido y apagado de diodos leds mediante pines digitales utilizando la herramienta LINX. Los elementos a necesitar para esta práctica son diodos led, resistencia y una tarjeta Arduino.

En la figura se muestra el diseño esquemático. El LabVIEW por medio de la herramienta LINX gestionará todos los datos digitales desde la interfaz diseñada, la misma que servirá como visualizador de las asignaciones discretas en los pines digitales previamente configurados.

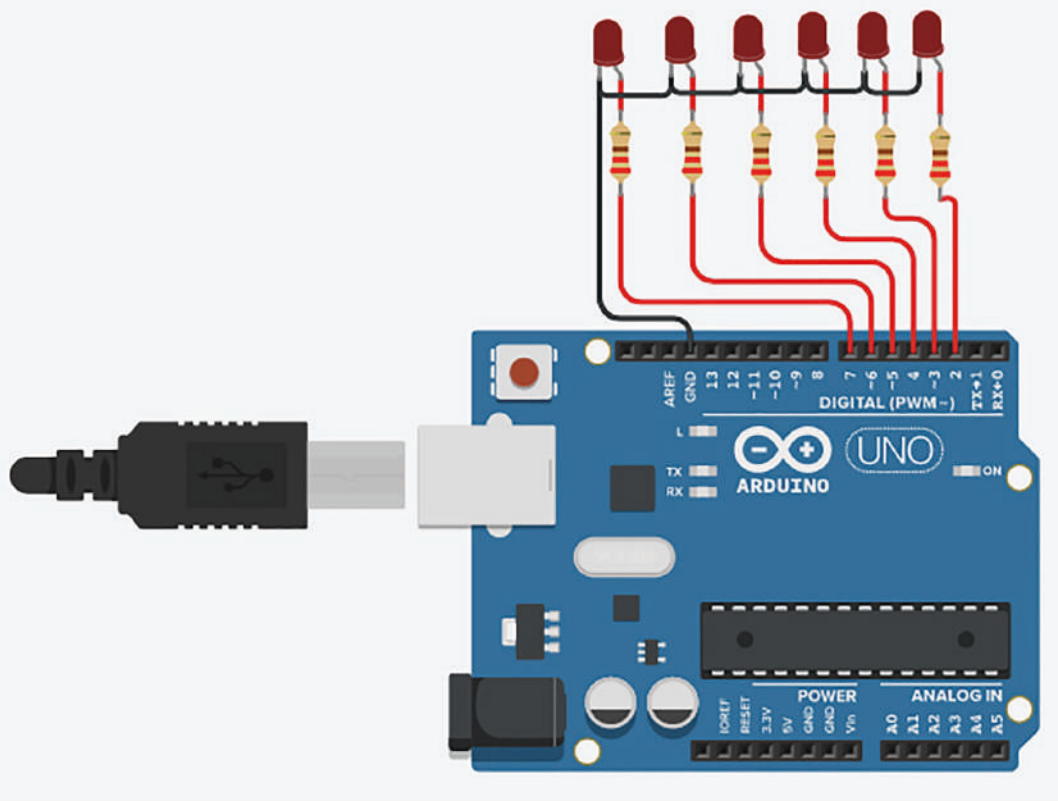



Figura 139. Esquemático del circuito para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

Se cargará un firmware prestablecido por la herramienta de LINX., ver figura. El cual se encargará de enlazar la interfaz desarrollada en LabVIEW y el Arduino. Teniendo el control de gestión de datos desde la interfaz realizada en LabVIEW.

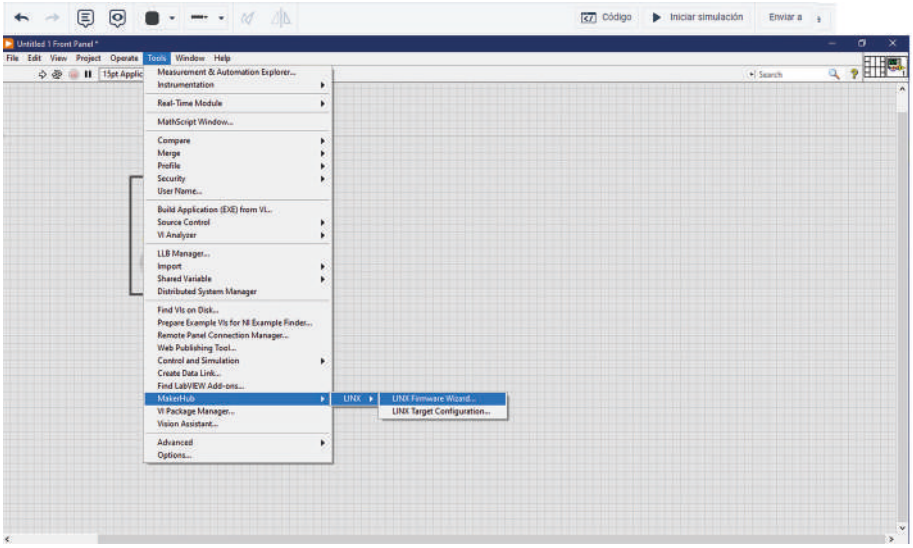



Figura 140. Herramienta de MakerHub LINX. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para el cargue del Firmware se selecciona la familia del sistema embebido, luego el tipo de la placa a utilizar y por último el método de cargue del firmware, ver figura.


Figura 141. Interfaz para cargar Firmware a sistema embebido. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Por último, se selecciona el puerto COM donde se encuentra conectado el sistema embebido y la forma de cómo se cagará el programa a la Arduino. Ya teniendo programada la placa se procede a la programación de la interfaz la cual va a gestionar los datos digitales.

A continuación, la figura muestra el diseño del software que enviará el dato de cada botón respectivamente.

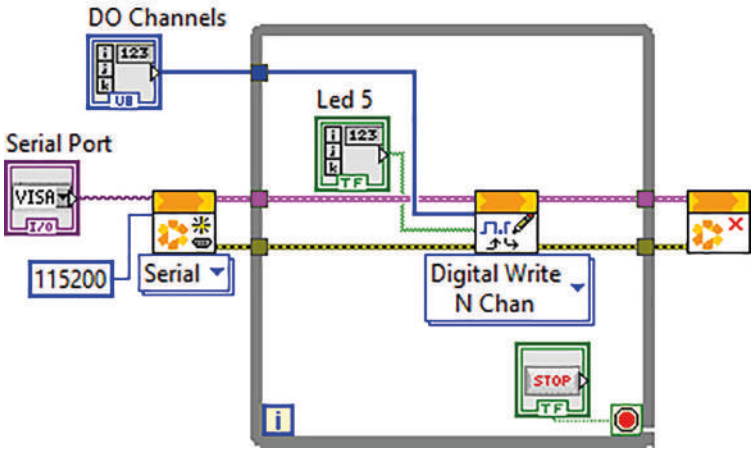



Figura 142. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al momento de la ejecución desde la interfaz de LabVIEW, se gestionan los valores de cada puerto del sistema embebido, ver figura, previamente configurados en la interfaz de LabVIEW.

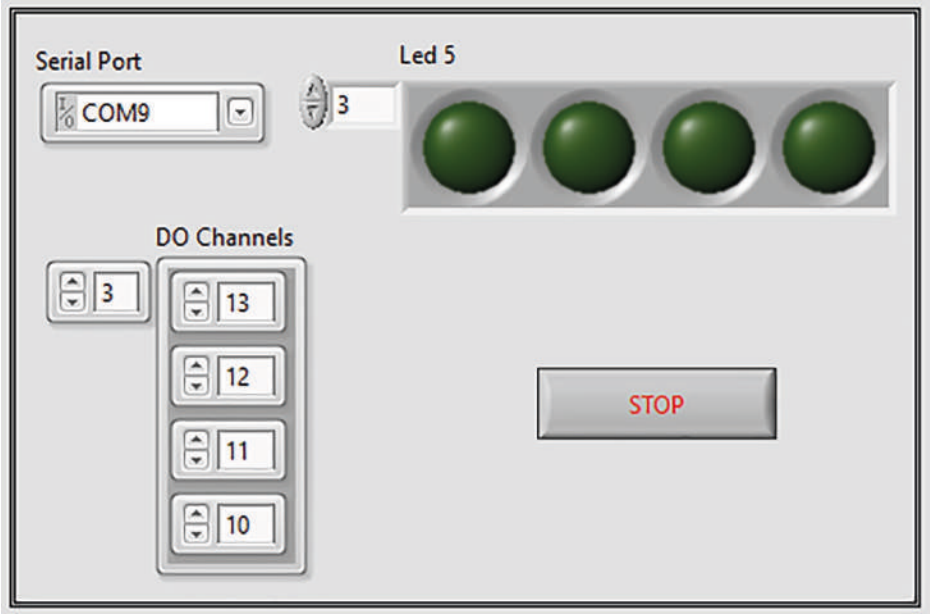



Figura 143. Panel frontal de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.3.3 **Practica #5. Lectura De Un Grupo De Entradas Digitales**

Para esta práctica se realizará la lectura del estado digital del sistema embebido, para un grupo de entradas digitales de una Arduino.

Para esta práctica, desde la interfaz digital desarrollada en LabVIEW se podrá seleccionar el grupo de pines digitales a utilizar y por medio de unos indicadores en la interfaz podríamos monitorear el estado de estos.

Para empezar, debemos configurar Arduino y lo hacemos poniendo el bloque “open Serial”, al que le configuramos el puerto y la tasa de comunicación, luego se procese a utilizar el bloque “Digital red”, para obtener los estados de cada una de las entradas digitales declaradas como se observa en la figura.

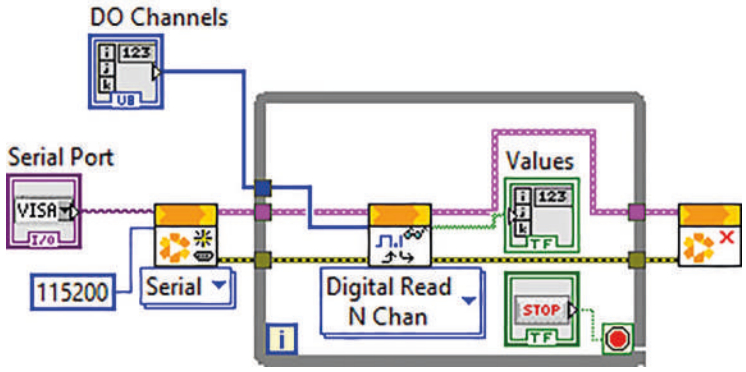



Figura 144. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al instante de ejecutar la interfaz desarrollada en LabVIEW, se reciben los estados de cada uno de los pines configurados como entradas digitales, ver figura.

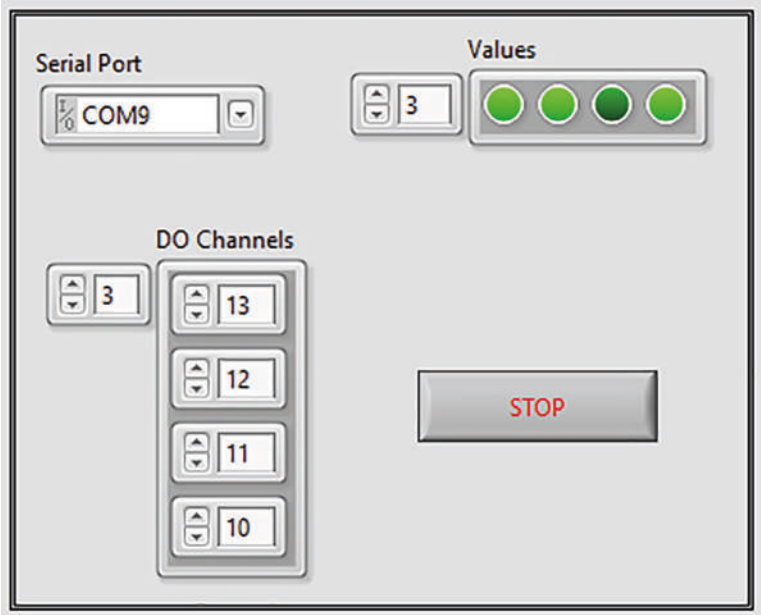



Figura 145. Panel frontal de instrumentación virtual de la aplicación. Panel frontal de instrumentación virtual de la aplicación. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.3.4 **Elaboración De Contador Ascendente Y Descendente**

Para esta práctica pondremos en práctica un contador ascendente y descendente, partiendo de dos datos de entrada digitales uno para incrementar el conteo y el otro para decrementar.

En la programación se detectan flancos en los cambios de estado de las señales digitales con el fin de llevar un conteo acorde a los pulsos de entrada, ver figura.

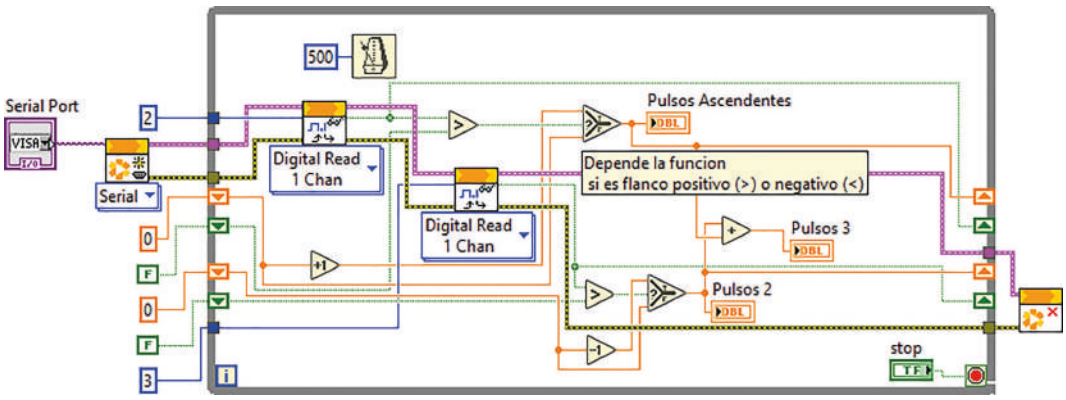



Figura 146. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la interfaz desarrollada en LabVIEW, se presentan los conteos ascendentes y descendentes y la diferencia de estos, ver figura.


Figura 147. Panel frontal de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Estos contadores dependen de las señales de entrada digitales provenientes de la electrónica, ver figura.

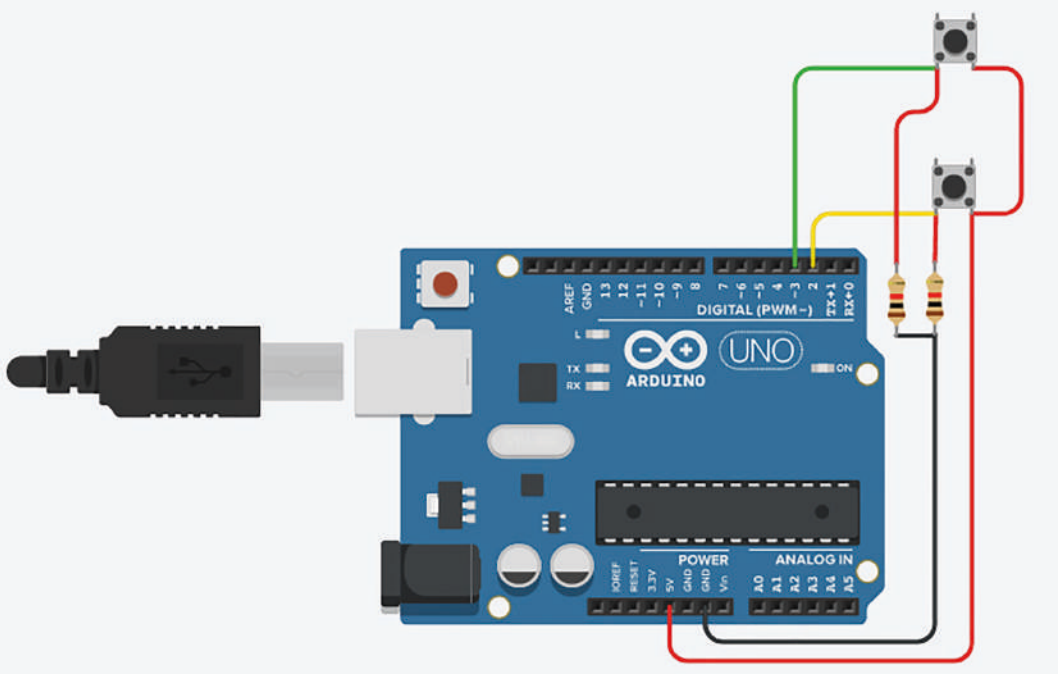



Figura 148. Esquemático del circuito para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

## 3.4 Entradas Y Salidas Analógicas

### 3.4.1 Practica #6. Adquisición Y Registro De Datos De Una Señal Analógica LM35 Con Gestión Desde La Arduino

Para esta práctica se desarrollará la lectura y comprensión de un sensor LM35 el cual tiene una resolución de 10 milivoltios por grado Celsius captado. El sistema embebido a utilizar es un Arduino Uno, el cual tiene un microcontrolador Atmega 328p que enviará el dato mediante la herramienta NI-VISA.

La figura se muestra el diseño esquemático en Tinkercad. Para esta práctica se entrelaza la Arduino con la interfaz LabVIEW y que servirá para enviar datos luego del procesamiento del mismo, y finalmente captar el valor de la temperatura obtenida por el sensor.

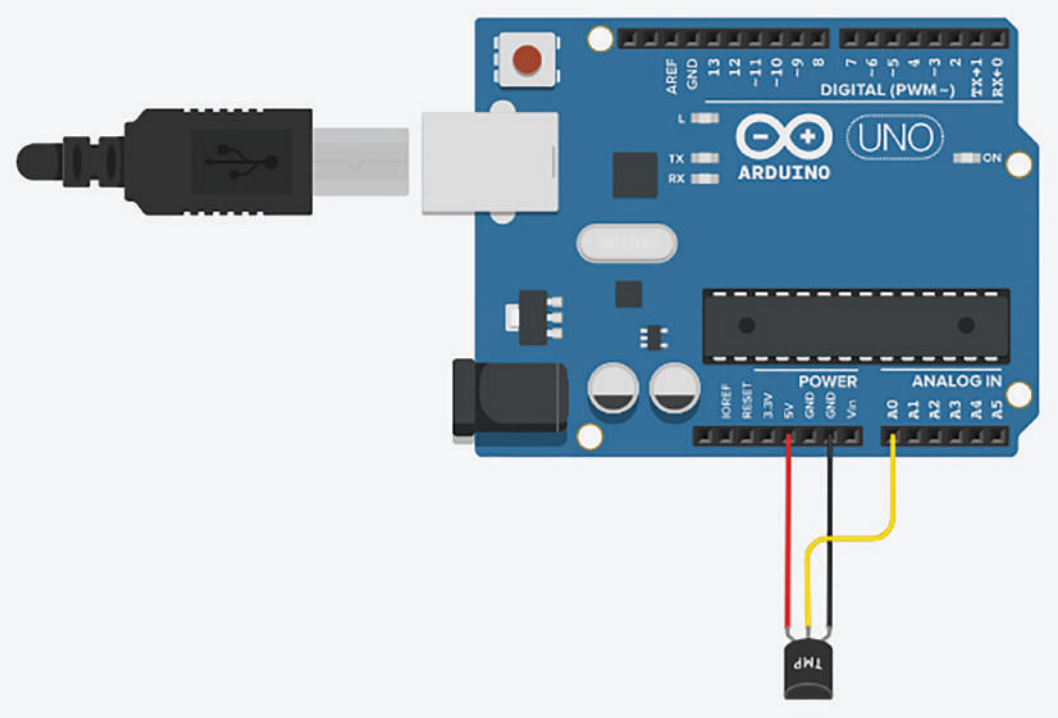



Figura 149. Esquemático de simulación en Isis Proteus para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

En el código de la Arduino primero se asigna variables como pin de entrada análoga y la variable donde se guarda dicho dato.

En el void setup se configurará la tasa de comunicación en baudios a trabajar, también la referencia de la conversión análoga del sensor es 10 milivoltios por grados, que se utiliza como referencia interna correspondiente al mismo microcontrolador.

En el void loop inicialmente se utiliza para obtener un dato mediante una conversión análogo digital, este será procesado con la fórmula que divide 1,1V sobre la resolución de 10 bits, cada paso en la lectura análoga es de aproximadamente 0.001074V = 1.0742 mV. Si 10mV es igual a 1 °C, entonces 10/1.0742=~9.31. Por lo tanto, para cada cambio de 9.31 en la lectura analógica, hay un grado de cambio de temperatura y enviado por medio del puerto COM hacia el panel frontal de LabVIEW.

```
float Temperatura;
int Sensor=A0;
void setup() {
analogReference(INTERNAL);
Serial.begin(9600);
}

void loop() {
  Temperatura=analogRead(Sensor);
  Temperatura=Temperatura/9.31;
  Serial.println(Temperatura);
  delay(400);
}
```

Figura 150. Código de programación de configuración de variables para aplicación. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

A continuación, se evidencia el diseño de la programación en bloques que se encargará de leer los datos, posteriormente se procederá a comprar la conversión con los valores constantes, así se busca visualizar una alarma siempre y cuando, la temperatura cumpla con las condiciones.

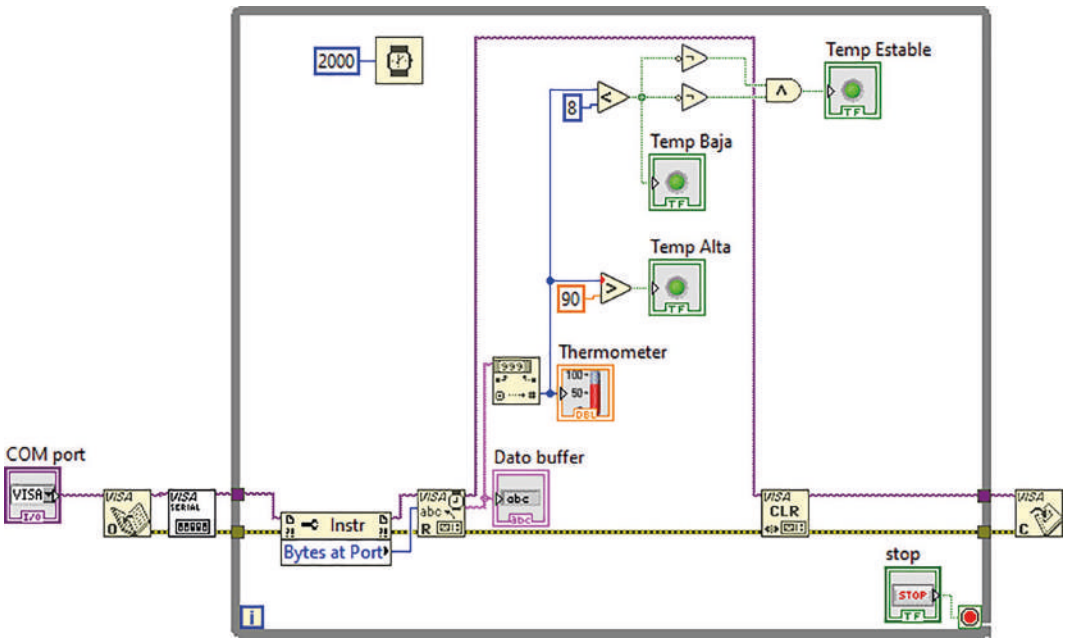



Figura 151. Programación de aplicación. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la figura se muestra la ejecución del panel de control diseñado en LabVIEW, que permite visualizar los datos y las condiciones previamente establecidas desde el microcontrolador Arduino hacia el puerto COM.

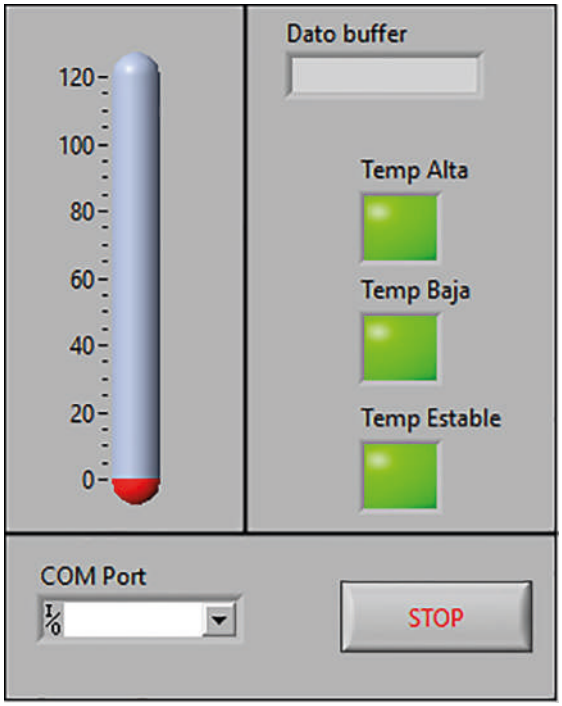



Figura 152. Panel frontal de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.4.2 **Practica #7. Adquisición Y Registro De Datos De Una Señal Analógica LM35 Con LINX**

Para esta práctica se desarrollará la lectura y comprensión de un sensor LM35 el cual tiene una resolución de 10 milivoltios por grado Celsius captado. El sistema embebido a utilizar es un Arduino Uno, el cual tiene un microcontrolador Atmega 328p.

La figura se muestra el diseño esquemático en Tinkercad. Para esta práctica se entrelazó la Arduino con la interfaz LabVIEW, por medio de la herramienta LINX de MakerHub para luego captar el valor de temperatura obtenida por el sensor.

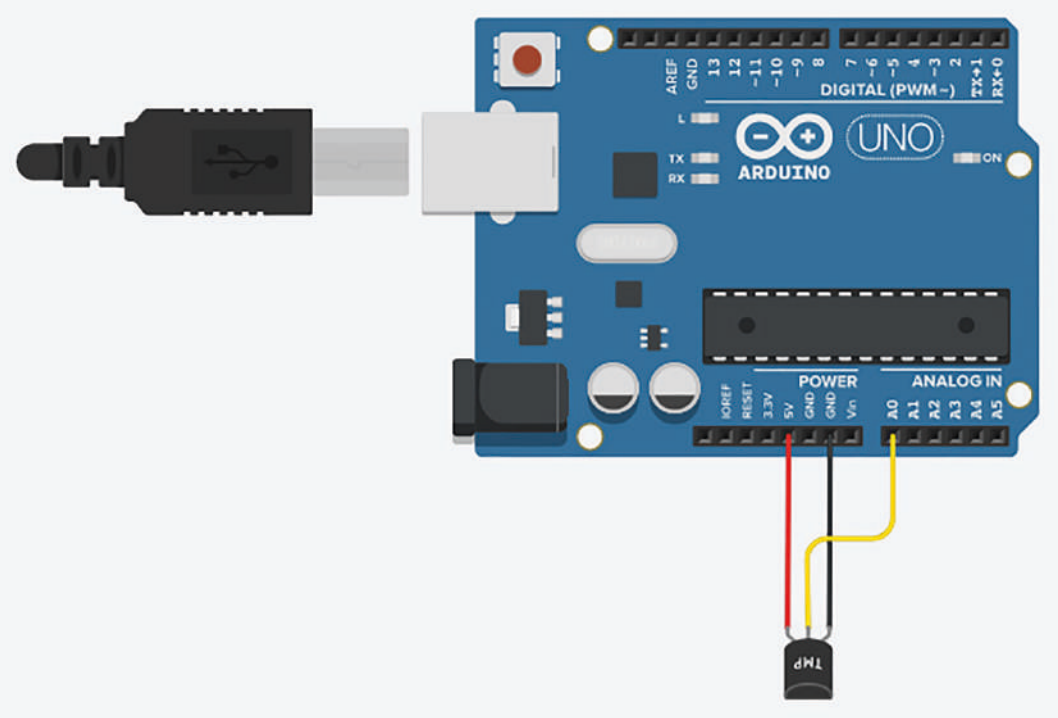



Figura 153. Esquemático de simulación en Isis Proteus para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

Para esta práctica cargaremos en la Arduino el firmware de LINX, el cual se encargará de enlazar el sistema embebido con el LabVIEW, centralizando la gestión de los datos directamente desde la interfaz.

A la hora de programar en la interfaz de LabVIEW, se utiliza en modulo TMP35 de LINX, ver figura. El cual se encarga internamente de configurar el pin de entrada de dato y de recibirla por el puerto análogo, para visualizarla en la interfaz.

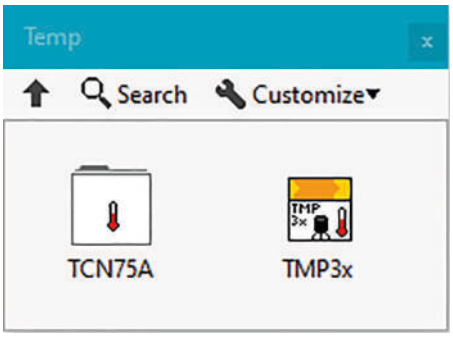



Figura 154. Modulo TMP3X del LINX - MakerHub. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El código desarrollado en LabVIEW en el cual se configura el COM donde se encuentra conectada la Arduino y de configura el módulo TMP35 encargado de parametrizar la entrada del sensor de temperatura se puede ver en la figura.

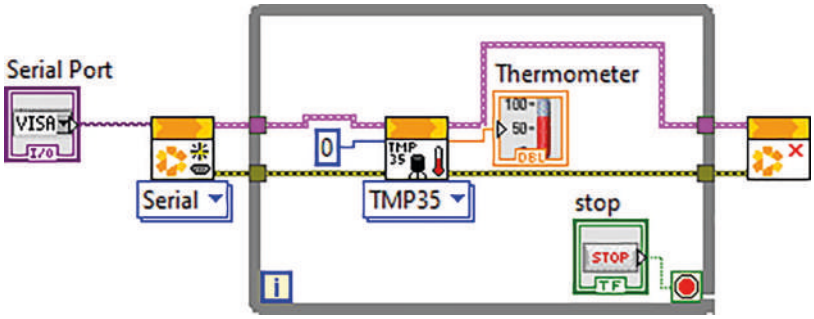



Figura 155. Código de programación de la interfaz web para la lectura de temperatura con sensor TMP35.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la figura se muestra el diseño del software de instrumentación virtual LabVIEW que se encargará de la gestión y visualización de la temperatura.

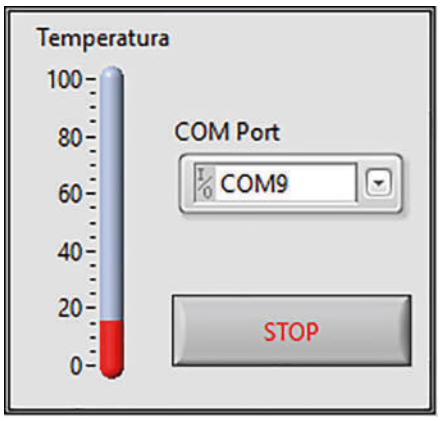



Figura 156. Panel frontal de la aplicación de medición de temperatura.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.4.3 **Practica #8. Control ON/OFF De Ventilador A Partir Del Censado De Temperatura Con LINX**

El fin de esta práctica es crear un programa a través del cual se puedan obtener lecturas de temperatura, así como controlar en forma automática un ventilador para poder controlar la ambientación.

Para iniciar con la práctica debemos de tener un ventilador DC, un sensor de temperatura y el sistema embebido como elementos electrónicos. Con el fin de sensar la temperatura y poder variarla por medio del ventilador, ver imagen.

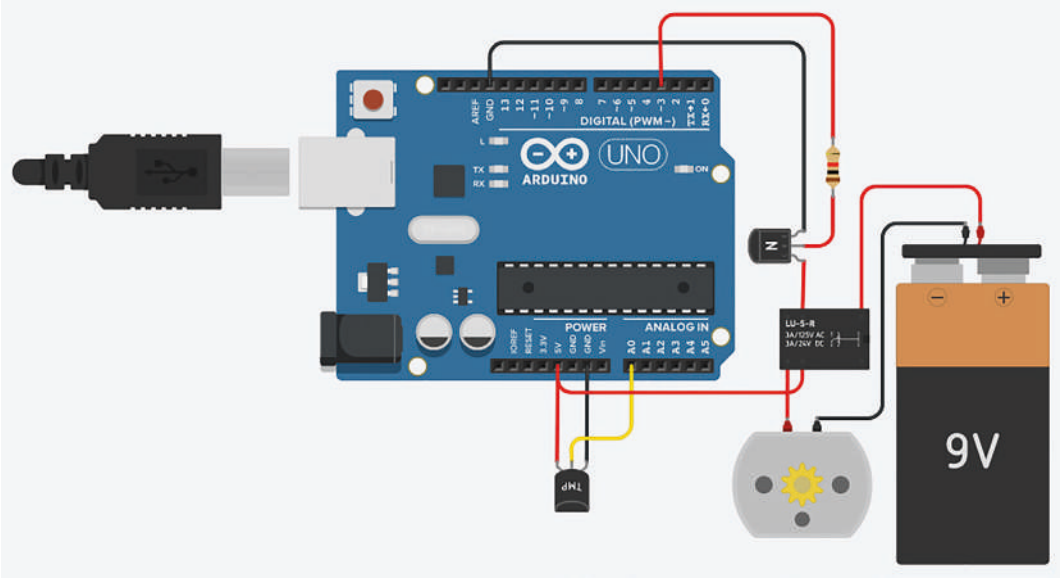



Figura 157. Conexión Sensor, Actuador y ventilador al sistema embebido. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

Desde el diagrama de bloques se va a trabajar con la herramienta LINX que nos permitirá configurar el pin de la entrada analógica de la Arduino, en el cual se conectará el sensor de temperatura. Por otro lado, se usará un pin digital para la activación o desactivación de un relé, para poder encender o apagar un ventilador DC, ver figura.

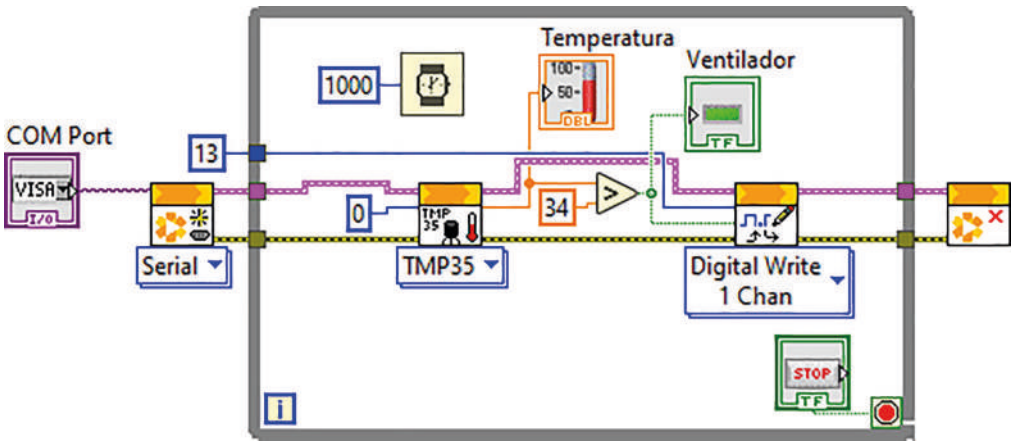



Figura 158. Diagrama a bloques para medir temperatura. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la figura anterior se puede observar que el dato de temperatura luego se ser sensado, se compara con una referencia para este caso de 34 grados Celsius para luego tomar la decisión de encender o apagar el ventilador DC.

La vista en el panel frontal es la siguiente:

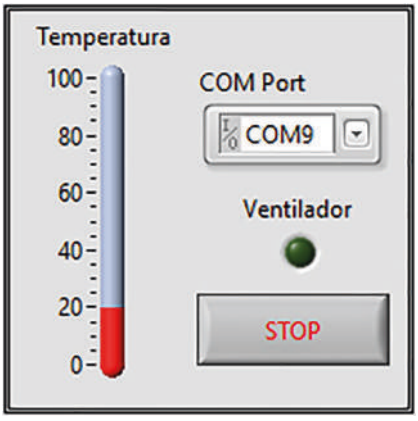



Figura 159. Panel frontal medición de temperatura. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.4.4 **Practica #9. Aplicación De Llenado De Tanque**

En la presente práctica asignaremos valores a un control-tank en la interfaz de LabVIEW. Se utilizará un microcontrolador Atmega 328p de Arduino que se encarga de la gestión. Enviará el dato mediante la herramienta NI-VISA, por medio del puerto COM que se configurará desde la interfaz.

En la figura se muestra el diseño esquemático en Tinkercad el cual detallará la electrónica a utilizar que se entrelaza con la interfaz en LabVIEW. La cual servirá para adquirir los datos del potenciómetro y su respectiva conversión análoga a digital.

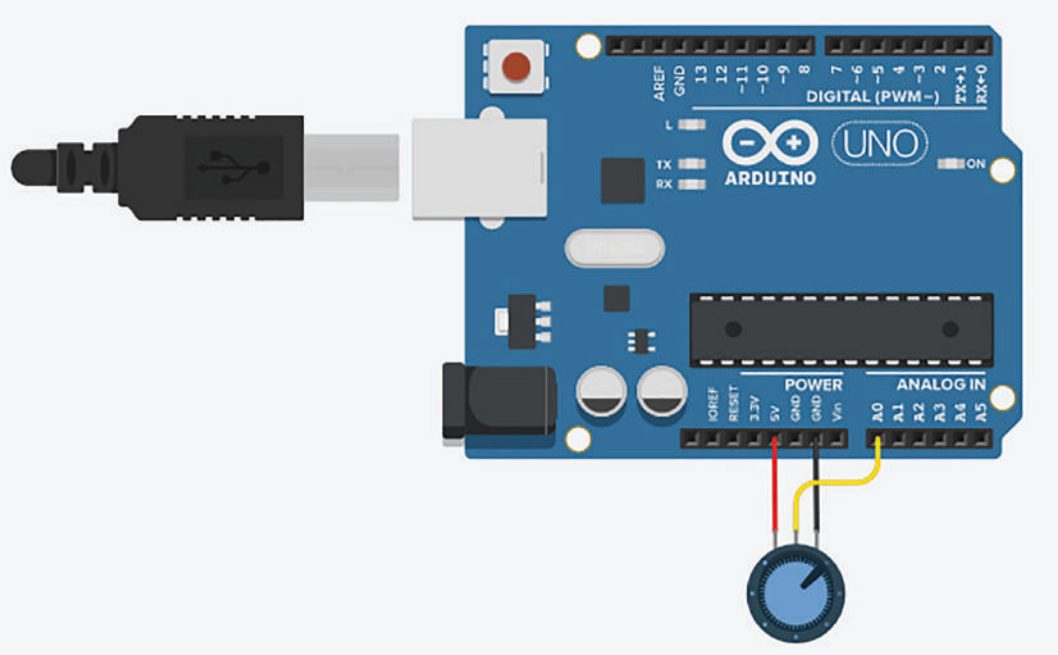



Figura 160. Esquemático de circuito electrónico para la aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

El código de la práctica, ver figura, obtendrá un dato mediante la conversión A/D, el mismo que será procesado y enviado por medio del puerto COM hacia el panel frontal de LabVIEW.

```
const int analogInpin=A0;
int DatoAnalogico=0;
void setup() {
  Serial.begin(9600);
}

void loop() {
  DatoAnalogico=analogRead(analogInpin);
  Serial.println(DatoAnalogico);
  delay(2);
}
```

Figura 161. Código para llenado de un tanque. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En la figura, se muestra la programación por bloques para el ejercicio, dicho código se encarga de leer los datos del buffer, dicho dato se convertirá un String a un dato numérico, para posteriormente ser apreciado por el slider bar.

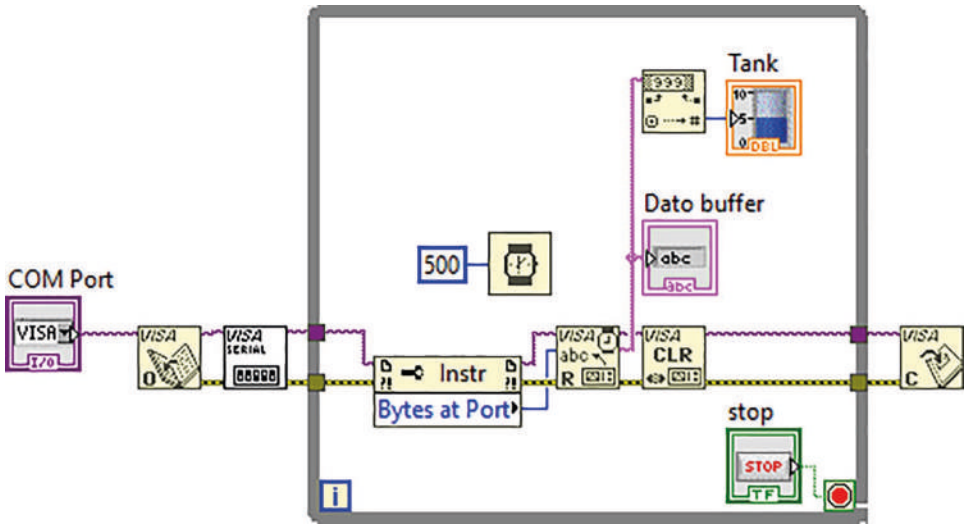



Figura 162. Programación para la aplicación práctica. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al momento de la ejecución del instrumento virtual diseñado, ver figura, se visualizará el dato enviado desde Arduino hasta el puerto COM.

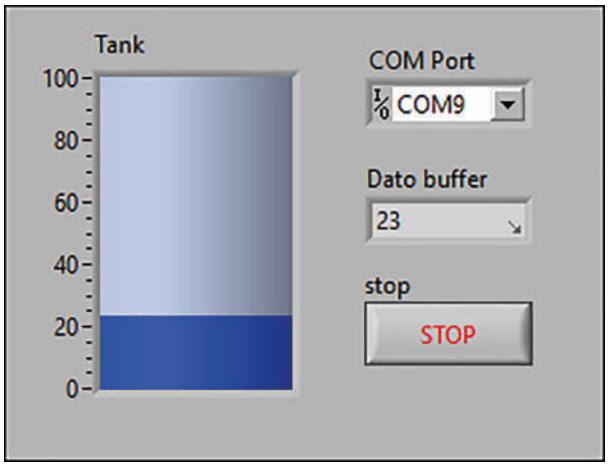



Figura 163. Panel frontal de instrumentación virtual de la aplicación práctica. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

## 3.5 Sensores Básicos De Luz, Distancia Y Presencia

### 3.5.1 Practica #10. Medida De Luz

Esta práctica consiste en desarrollar una aplicación para medir la cantidad de luz en diferentes ambientes.

Para ello recurrimos a un bloque de función de la LINX de MakerHub que se encarga de llevar esta tarea. Bloque “Phocell”

Los Parámetros que hemos de configurar en este bloque se muestra n la figura.

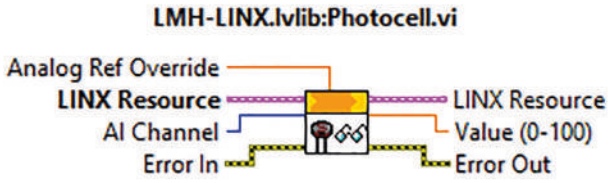



Figura 164. Bloque Photocell de LINX. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El pin a la que conectaremos la fotocélula “AI Channel”, la tensión de referencia máxima que colocamos en este montaje, la salida del bloque es el valor equivalente a la luz medida comprendido entre 0 y 100. El código de la interfaz en LabVIEW se puede ver en la figura.

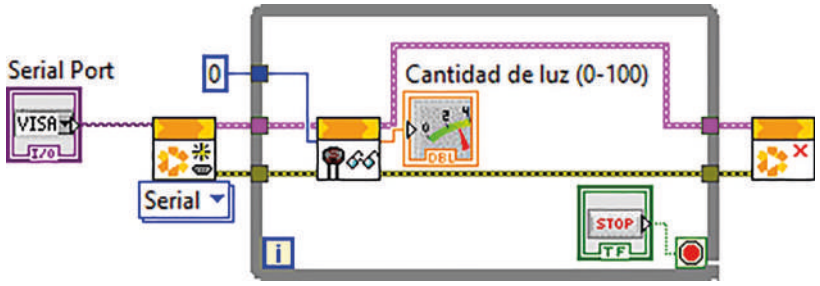



Figura 165. Diagrama de bloques de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La salida del bucle, para este montaje se usará mediante un botón “Stop”, la figura 165 muestra la interfaz gráfica

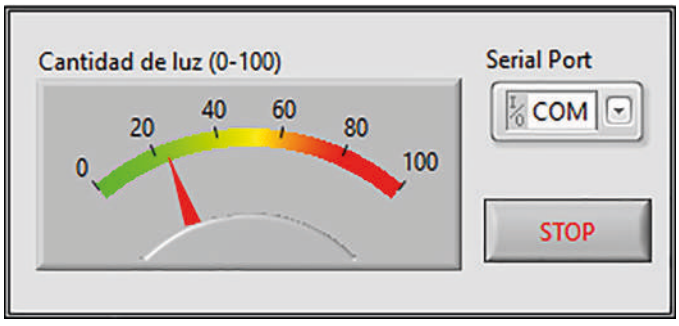



Figura 166. Panel frontal de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En el montaje del circuito solo se necesita un divisor de tensión conformado por una resistencia y una photoresistor, junto al sistema embebido, ver imagen.

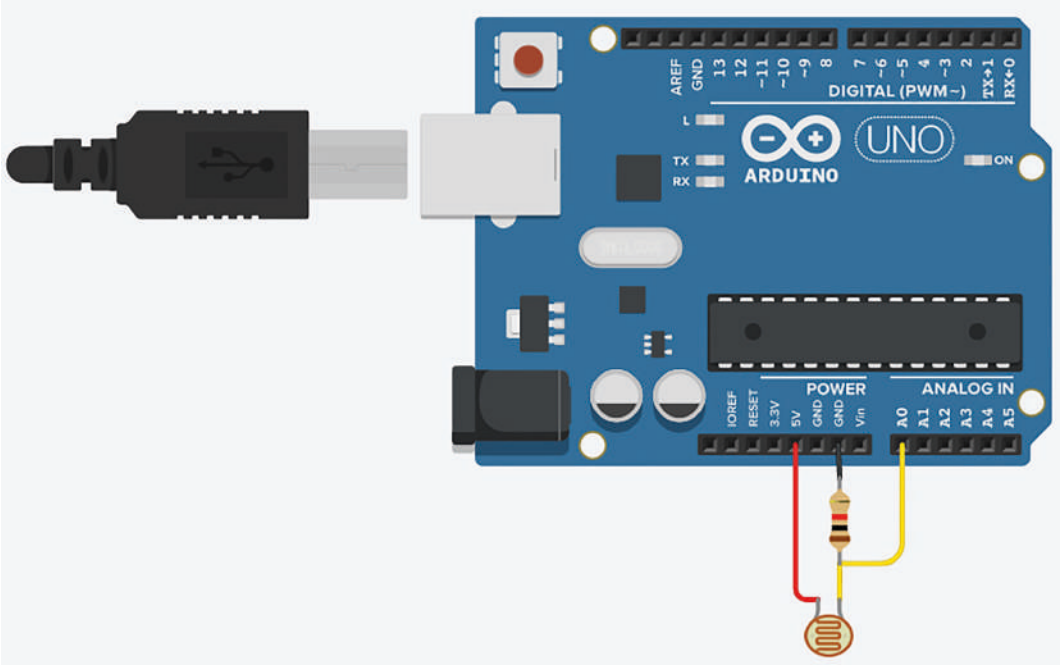



Figura 167. Diagrama del circuito del circuito medidor de luz. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

### 3.5.2 **Practica #11. Medición De Distancia**

Esta práctica consiste en desarrollar una aplicación para medir distancia de un objeto.

Para ello recurrimos a un bloque de función de la LINX de MakerHub que se encarga de llevar esta tarea. Bloque “Ultrasonic”

Los Parámetros que hemos de configurar en este bloque se muestra n la figura.

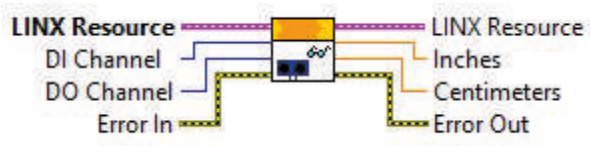



Figura 168. Bloque Ultrasonic de LINX. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al boque de Ultrasonic se le configura un pin de entrada y un pin de salida a la que conectaremos el sensor de “Ultrasonido”, a la salida del bloque podremos obtener la distancia en pulgadas o en centímetros. El código de la interfaz en LabVIEW se puede ver en la figura.

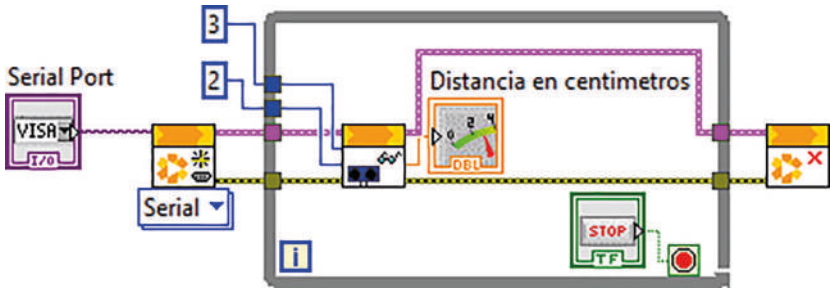



Figura 169. Diagrama de bloques de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La salida del bucle, para este montaje se usará mediante un botón “Stop”, la figura 165 muestra la interfaz gráfica

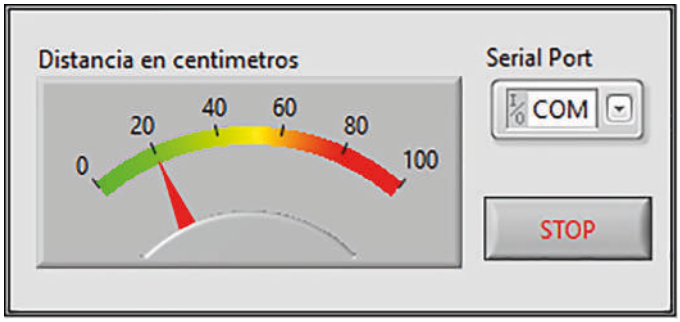



Figura 170. Panel frontal de la aplicación. Autoría Propia
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En el montaje del circuito solo se necesita un divisor de tensión conformado por una resistencia y una photoresistor, junto al sistema embebido, ver imagen.

## 3.6 Control De Motores DC, Paso A Paso, Servos, Brushless

### 3.6.1 Practica #12. Control De Motor DC Con Gestión Desde La Arduino Y La Librería Del LINX

Esta práctica tendrá el objetivo de asignar valores discretos para lograr así, controlar un motor DC utilizando las herramientas de NIVISA. En la figura 170se muestra el esquemático electrónico que interviene en la implementación, pero que nos mostrara el control eficaz del mismo.

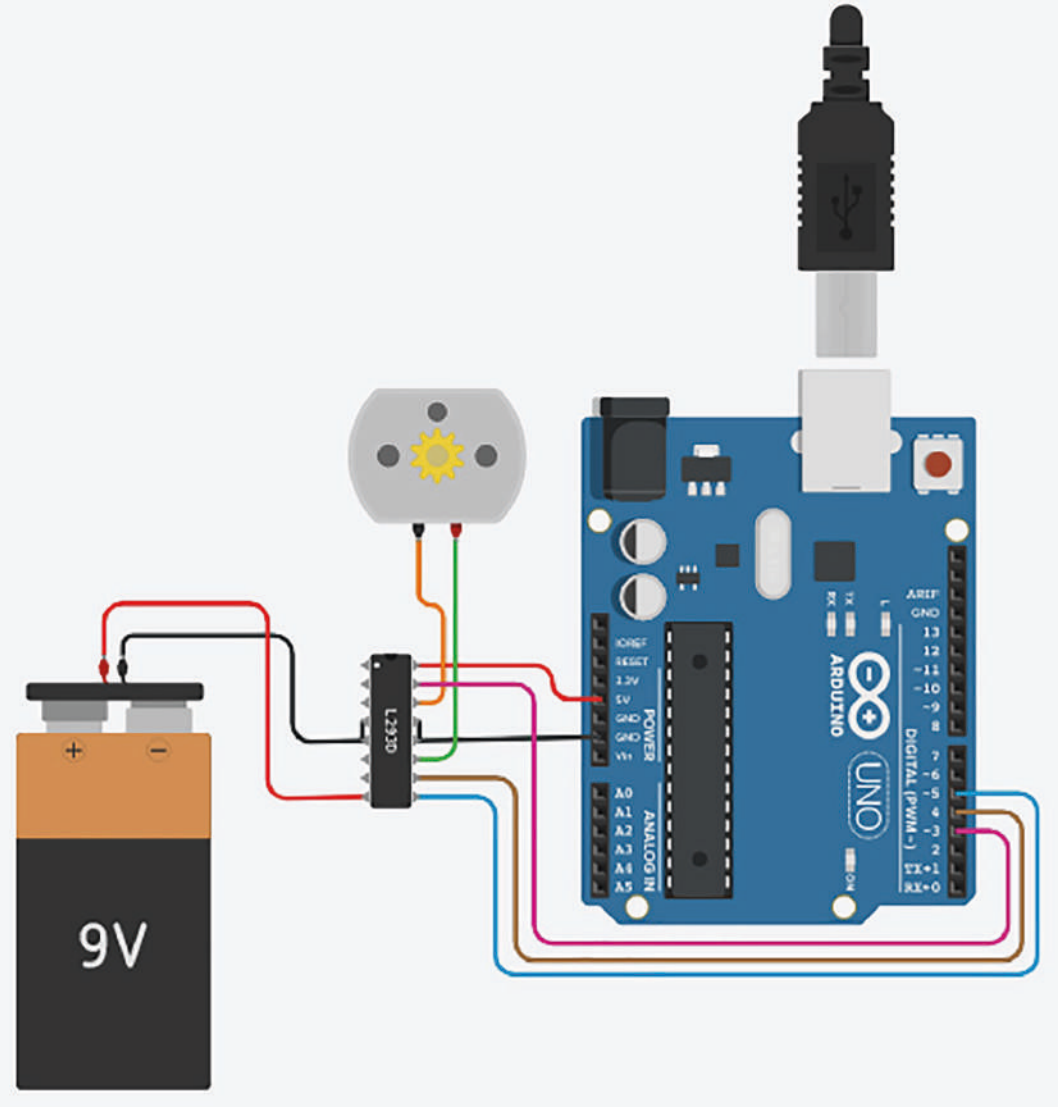



Figura 171. Esquemático de simulación en Isis Proteus para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

En la siguiente figura, se presenta un planteamiento para la secuencia de las funciones que se utilizarán en el control de movimiento del motor, estos algoritmos se plantean en el lenguaje del entorno del desarrollo de Arduino, en el entorno de desarrollo integrado IDE de Arduino. Se realizará un mapeo del dato ubicado en el buffer del puerto COM y lo envía a LabVIEW.

La programación en bloques de LabVIEW está diseñada y se muestra en la figura 171, permitirá mapear el estado actual de los botones y enviar comandos mediante NI VISA hacia el puerto COM.

```
char datoBuffer;
int a1=3;
int a2=4;
int en=5;
void setup() {
  Serial.begin(9600);
  pinMode(a1,OUTPUT);
  pinMode(a2,OUTPUT);
  pinMode(en,OUTPUT);
}
void loop() {
  while(Serial.available()){
    datoBuffer=Serial.read();
    if(datoBuffer='A'){
      digitalWrite(a1,HIGH);digitalWrite(a2,LOW);
      }
    else if(datoBuffer='B'){
      digitalWrite(a1,LOW);digitalWrite(a2,HIGH);
      }
    else if(datoBuffer='C'){
      digitalWrite(a1,LOW);digitalWrite(a2,LOW);
      }
    else if(datoBuffer='D'){
      digitalWrite(en,HIGH);
      }
    else if(datoBuffer='E'){
      digitalWrite(en,HIGH);
      }
    }
  }
```

Figura 172. Código de control de motor DC para la aplicación.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

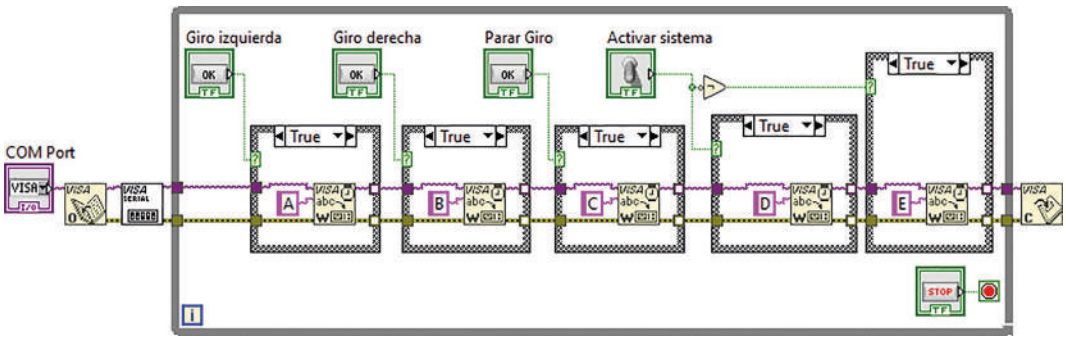



Figura 173. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Cuando se ejecuta el programa se capta el estado de los botones en el panel frontal, tal como se puede ver en la figura 173, el cual enviará datos al sistema embebido que interpreta los caracteres y se visualizará en el movimiento del motor.


Figura 174. Panel de simulación del instrumento virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La práctica anterior desde el punto de vista del desarrollo en LabVIEW puede ser un poco más sencilla haciendo el uso de la herramienta LINX de MakerHub. Para este caso el diagrama electrónico será igual al de la figura, contamos con la ventaja de no programar el sistema embebido por independiente, para este caso se cargará el firmware procedente del LINX como se ha realizado en prácticas anteriores.

Seguidamente se programará la interfaz o panel frontal en el LabVIEW, en donde la programación se hace más sencilla como se puede ver en la figura.

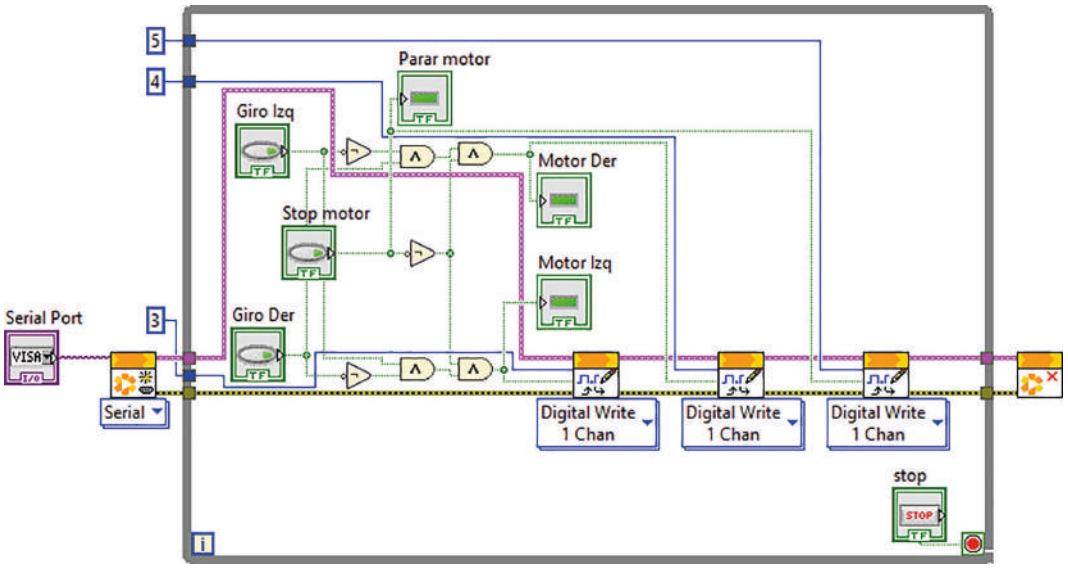



Figura 175. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Para esta aplicación desarrollada por medio de la herramienta LINX de MakerHub, la interfaz del LabVIEW sería la planteada en la figura.


Figura 176. Interfaz para el control de giro de motor DC. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.6.2 **Practica #13. Generación De PWM Para Control De Velocidad De Motor DC**

Esta práctica consiste en el control asíncrono de la velocidad de un motor DC, mediante la modulación por ancho de pulsos PWM y con la interfaz NI VISA. En la figura 176 se muestra el diseño electrónico para esta práctica. Dicho sistema estará entrelazado con la interfaz de LabVIEW logrando enviar el PWM para controlar la velocidad del motor DC. Para lograr el control de velocidad del motor, se tendrá un puerto COM el cual estará enlazado con la interfaz de LabVIEW, logrando así enviar el pulso para controlar la velocidad del motor DC.

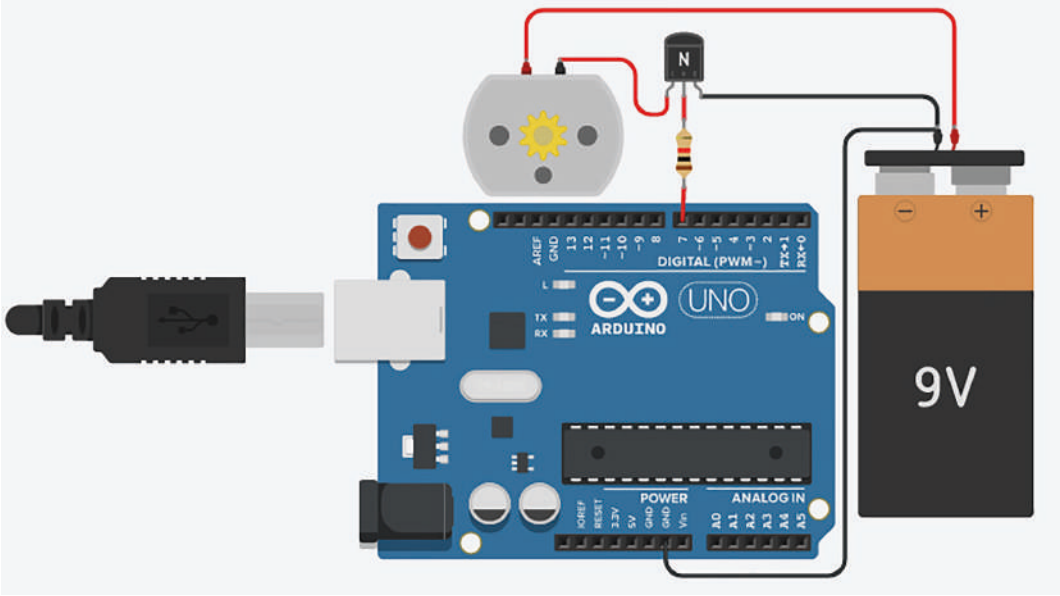



Figura 177. Esquemático de simulación para aplicación. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

El código de programación para esta practica se realiza con el lenguaje C, como se puede apreciar en la figura 177, se obtendrá el valor enviado por LabVIEW y se realiza la escritura utilizando el PWM con el fin de controlar la velocidad del motor mediante el ancho de pulso.

```
char datoBuffer;
int pwmpin=3;

void setup() {
  Serial.begin(9600);
  pinMode(pwmpin,OUTPUT);
}

void loop() {
  while(Serial.available()){
    datoBuffer=Serial.read();
    analogWrite(pwmpin,(unsigned int)datoBuffer);
    }
}
```

Figura 178. Código de control de motor DC usando PWM para la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

A continuación, en la figura 178 se muestra la programación en bloques y el panel en la figura 179 desarrollado en LabVIEW. Esta aplicación se encargará de enviar datos suministrados por una slider, mediante la herramienta VISA y dicho dato permitirá controlar la velocidad del motor mediante PWM.

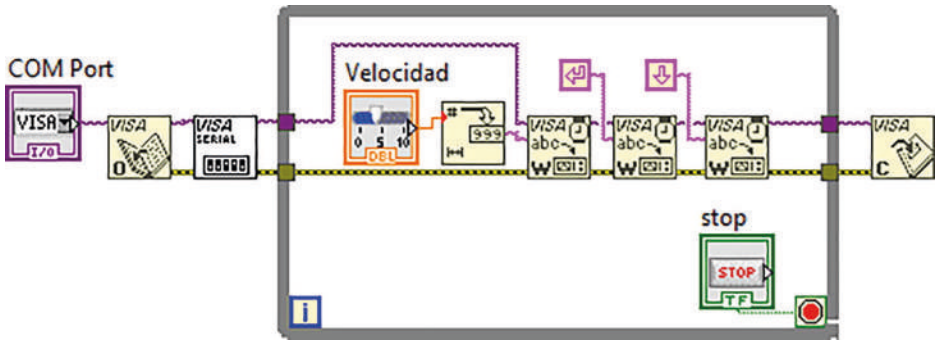



Figura 179. Programación en bloques para la practica 13. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Al momento de ejecutar el instrumento virtual diseñado, ver figura, servirá como maestro el cual envía los datos desde slider hacia el dispositivo Arduino y este ejecuta el control de la velocidad del motor DC.

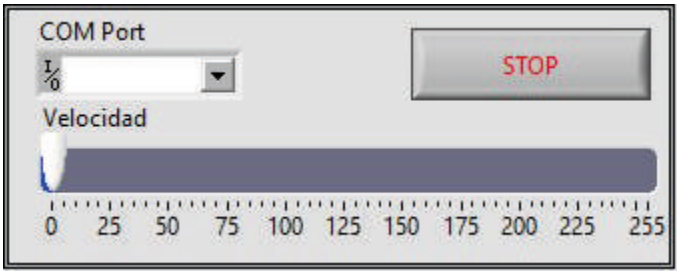



Figura 180. Panel Frontal de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Otra forma de realizar la interfaz del panel frontal en LabVIEW, es utilizando la herramienta LINX de MakerHub, para este caso la electrónica a implementar es la misma, lo que cambia en este caso es la programación en el LabVIEW, primero se cargara el firmware del LINX. Seguidamente, utilizaremos el bloque de PWM Set Duty Cycle, ver figura.

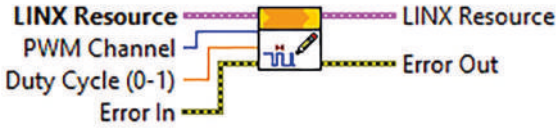



Figura 181. Bloque PWM Set Duty Cycle. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El diagrama de bloques a programar se puede visualizar en la figura.

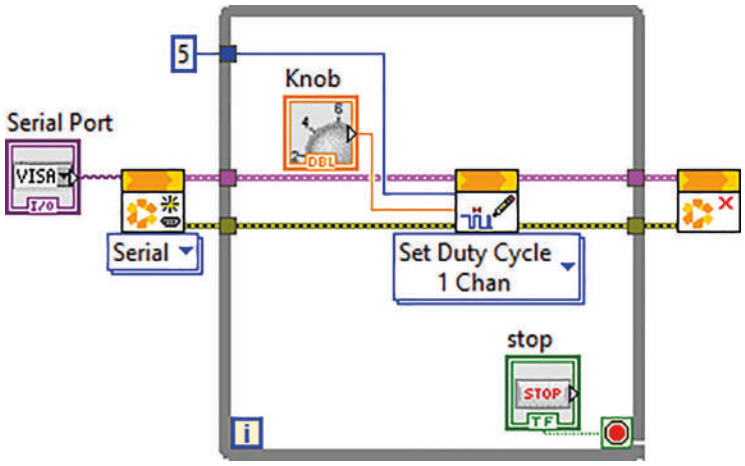



Figura 182. Programación de instrumentación virtual de la aplicación. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

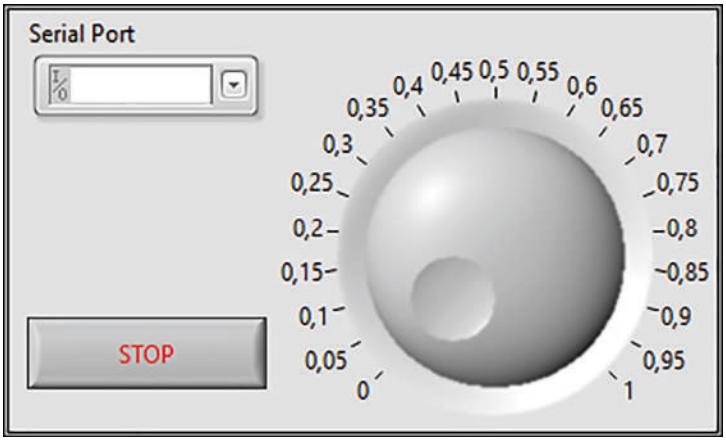



Figura 183. Panel frontal para el control de velocidad de motor DC con LINX. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.6.3 **Practica #14. Control De Motor Paso A Paso**

Los motores paso a paso son ideales cuando se requieren movimientos muy precisos. La característica principal de estos motores es que se pueden mover un paso a la vez por cada pulso que se le aplique. Este paso puede variar en función de la generación de micropasos del driver A4988. Estos motores pueden quedar enclavados en una posición o sin energizar.

Para esta práctica utilizaremos la librería LINX de MakerHub, Pero antes que todos entraremos en detalle con los componentes electrónicos del circuito que permite el movimiento del motor paso a paso, ver figura.

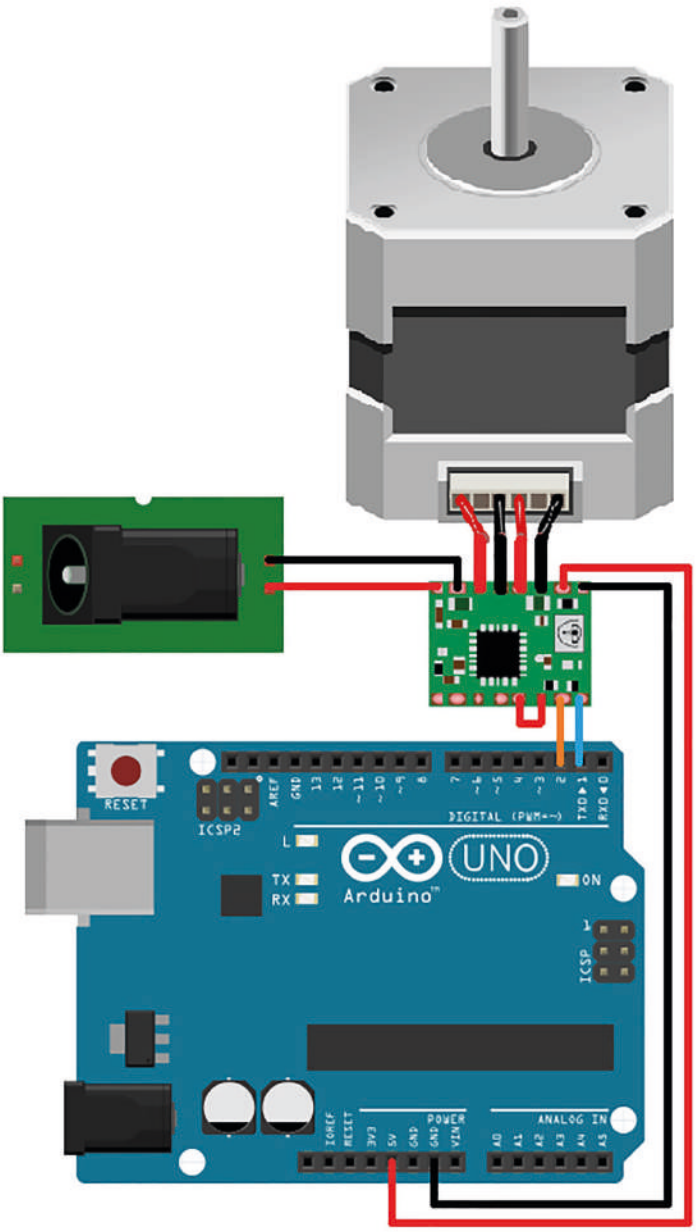



Figura 184. Circuito de control de motor paso a paso.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

Para esta práctica utilizaremos la librería LINX de MakerHub, Pero antes que todos entraremos en detalle con los componentes electrónicos del circuito que permite el movimiento del motor paso a paso, ver figura.

Ya teniendo el circuito montado se procede a cargar el firmware del LINX el cual permitirá adquirir datos de sistema embebido y procesarlo en el programa de LabVIEW. Teniendo el firmware cargado se programa el control del motor paso a paso, teniendo en cuenta el modo de funcionamiento del driver A4988. Por ende, se configurarán dos pines de salida digital, los cuales se encargarán de controlar la dirección o sentido de giro del motor y el otro se encarga de generar un tren de pulsos para generar los micropasos como se aprecia en la figura.

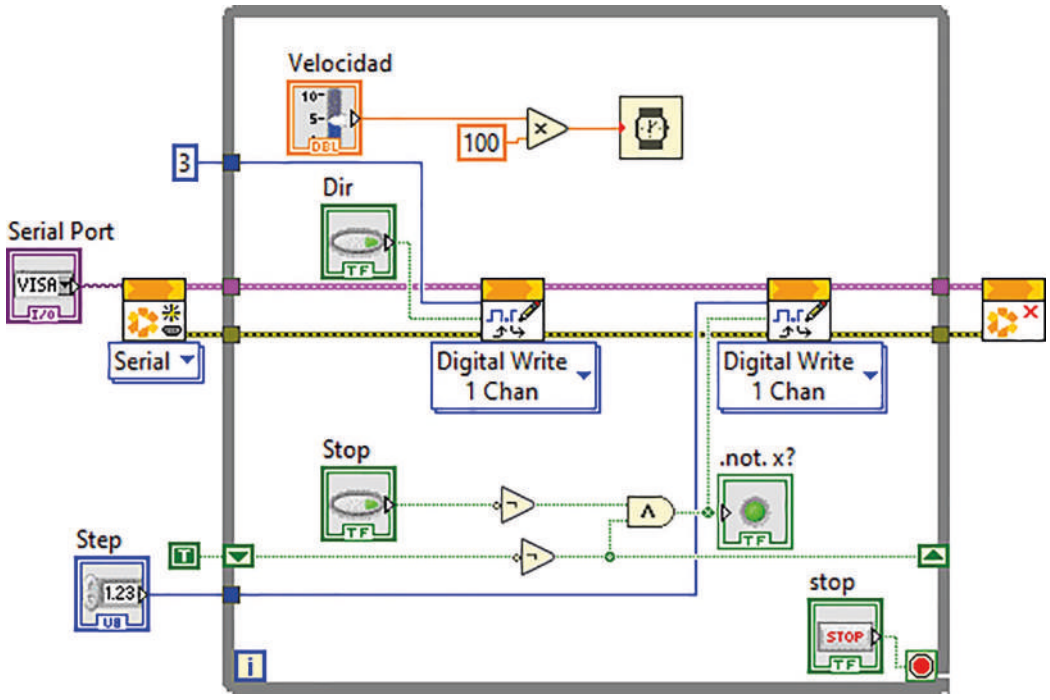



Figura 185. Código en LabVIEW para el control del motor paso a paso. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En el panel frontal, se contará con tres botones uno de control de sentido de giro del motor, otro para para el motor y un último para salir del programa, a su vez tenemos un control para el ingreso del puerto COM y otro control para la configuración del pin que generará el tren de pulsos, ver figura.


Figura 186. Panel frontal para el control de motor paso a paso. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

### 3.6.4 **Practica #15. Control De Servomotores**

Esta práctica se desarrollará una aplicación en LabVIEW para controlar el movimiento de un servomotor simple. La aplicación de esta práctica presenta diversas aplicaciones en muchas disciplinas que permitir a al estudiante entender y aprender sobre un dispositivo de control ampliamente usado en equipo industrial.

En función de la aplicación, la electrónica está comprendida un sistema embebido, servomotor, cables y conectores, ver figura.

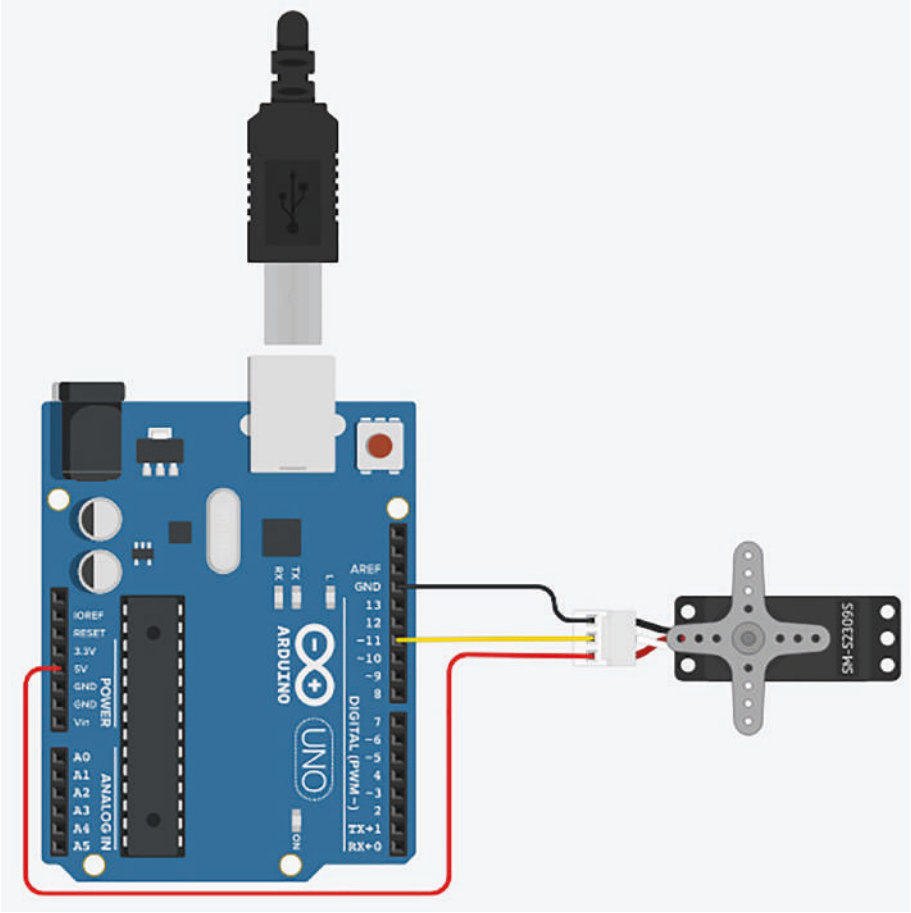



Figura 187. Circuito electrónico para el control de servomotor. Autoría Propia.

*Nota: Las imágenes fueron usadas en el software Tinkercad, de la suite Autodesk, bajo la licencia estudiantil otorgada por Autodesk.*

Un servomotor es un tipo de motor de corriente directa que puede posicionar su eje en cualquier ángulo dentro de un rango de operación determinado. La posición del servomotor se controla mediante una señal cuadrada de voltaje que indica su posición angular. La duración de la señal en el nivel encendido determina el ángulo de posición del motor.

Para lograr el control de un servomotor se utilizará la herramienta Servo Set Pulse Widch one channel, ver figura.

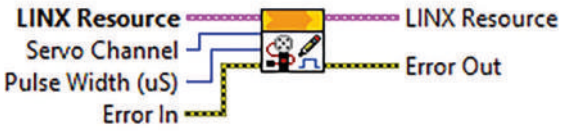



Figura 188. Bloque para control de servomotor de LINX. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El diagrama de bloque de la aplicación en LabVIEW se desarrolla como se presenta en la figura. Se puede apreciar que además de los bloques de comunicación se presentan dos bloques de configuración para iniciar el proceso del control del servo y otro para finalizar el control del servo, como se evidencia en la figura.

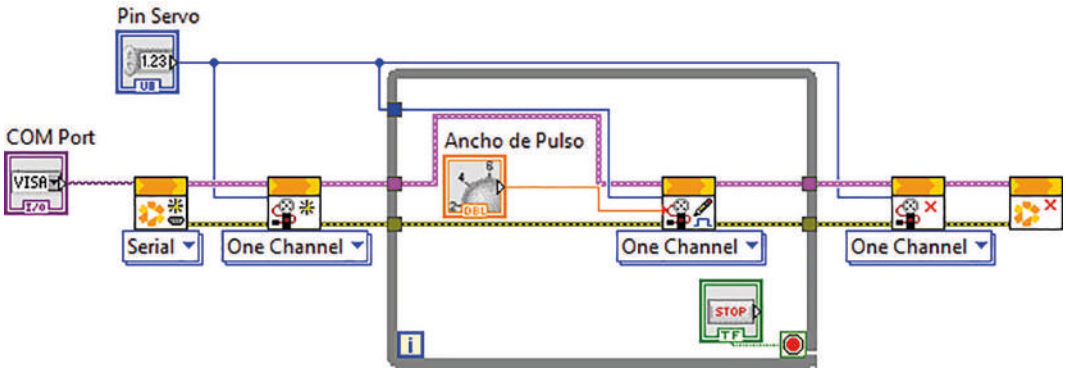



Figura 189. Diagrama de bloques para el control de un servomotor. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

En el panel principal de LabVIEW, se contará con el control que permite la configuración del COM, un control de ingreso para configurar el pin por donde se va a generar la modulación por ancho de pulso y una perilla Knob la cual permite variar el ancho de pulso como se observa en la figura.

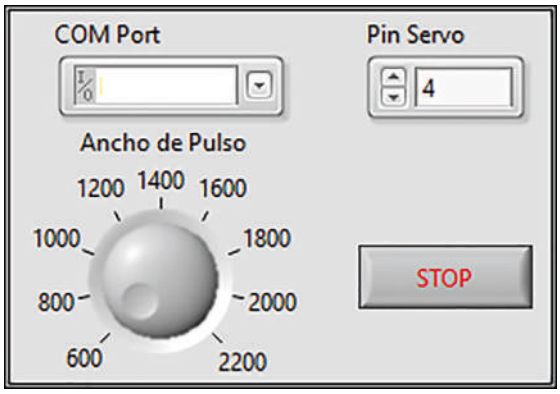



Figura 190. Panel frontal para el control de servomotor. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

# CAPÍTULO 4

## 4.1 LabVIEW Y El Internet De Las Cosas

El Internet de las cosas (IoT) se refiere a la interconexión de objetos cotidianos, tales como dispositivos electrónicos, electrodomésticos, maquinaria industrial y otros objetos físicos, a través de Internet. Estos objetos están equipados con sensores, dispositivos de almacenamiento y procesamiento de datos, y capacidad de comunicación que les permite interactuar y compartir información con otros dispositivos conectados a Internet.

La interconexión de estos objetos a través de Internet les permite recolectar y transmitir información en tiempo real. Esta información puede ser analizada y utilizada para tomar decisiones informadas, mejorar la eficiencia y reducir costos. El IoT tiene un enorme potencial en muchos campos, incluyendo la salud, la energía, la agricultura, el transporte, la manufactura y el hogar inteligente.

En la actualidad, el Internet de las cosas (IoT) está transformando la manera en que interactuamos con el mundo que nos rodea. Desde dispositivos conectados en el hogar hasta vehículos autónomos y equipos industriales inteligentes, el IoT se está convirtiendo en una parte cada vez más importante de nuestra vida cotidiana. En este contexto, el software LabVIEW se puede ser considerado una herramienta esencial para desarrollar aplicaciones IoT.

Como mencionamos anteriormente LabVIEW es un software de programación gráfica utilizado en todo el mundo para la automatización de pruebas, el control de procesos, la supervisión y el análisis de datos. Con su entorno visual de programación y su amplia variedad de herramientas integradas, LabVIEW es una opción para la creación de soluciones IoT. La combinación de LabVIEW y el IoT permite crear sistemas inteligentes capaces de interactuar con el mundo físico y recopilar datos valiosos en tiempo real.

La importancia del IoT radica en su capacidad para conectar dispositivos y sistemas en red, lo que permite la recopilación y el análisis de datos en tiempo real. Esto tiene implicaciones significativas en una amplia variedad de campos, desde la industria

manufacturera hasta la atención médica y la agricultura. La recopilación y análisis de datos en tiempo real pueden mejorar la eficiencia, reducir costos, prevenir problemas antes de que ocurran y, en última instancia, mejorar la calidad de vida de las personas.

Teniendo en cuenta la importancia del IoT y de la posibilidad de implementar herramientas con LabVIEW, abordaremos durante este capítulo protocolos que permiten esta conexión, este capítulo se abordará de manera teórica, ya que implementar prácticas de este tipo dentro de las aulas, tiene un mayor grado de dificultad, sin embargo, con la teoría evidenciada en este capítulo se dejará la posibilidad de abordar ejercicios prácticos diseñados por el docente o en su defecto, por el estudiante que quiera investigar más en el tema, siendo una opción recomendable para todos aquellos que desean realizar un proyecto orientado al IoT.

## 4.2 MQTT


*Figura 191 Logotipo Protocolo MQTT Tomada de Industrial Shields. 2020*

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería ligero diseñado para comunicaciones de máquina a máquina (M2M) y del Internet de las cosas (IoT). Fue creado en 1999 por Andy Stanford-Clark y Arlen Nipper de Arcom con el objetivo de proporcionar una forma eficiente y confiable de transmitir datos en redes limitadas en ancho de banda o poco confiables.

MQTT utiliza un modelo de publicación/suscripción, en el cual los dispositivos conectados a la red envían mensajes a través de un intermediario, conocido como Broker MQTT. Los mensajes pueden ser de varios tipos, como datos de sensores, comandos de control, alertas, entre otros.

Una forma de entender sencillamente la razón o importancia del protocolo, es relacionarlo con un ejercicio práctico, por ejemplo, el MQTT Client podría ser un sensor de temperatura que tenemos instalado en un invernadero, este sensor tomará el valor de la temperatura y lo enviará al MQTT breaker, que es conocido como el intermediario, quien, a su vez, enviará esta información hacia un computador, una Tablet o un celular.

Esto es un ejemplo sencillo, la grandeza y dificultad de las aplicaciones dependerá del interés que haya por desarrollar aplicaciones IoT, estas aplicaciones podrán ser de gran ayuda como un proyecto de una materia, de una carreara o de algún título de postgrado, por lo que recomendamos estas aplicaciones y su respectiva investigación.

### 4.2.1 **Instalar Protocolo MQTT**

La instalación de este protocolo no viene por defecto en la instalación del software, así desde un inicio se seleccionen todos los complementos y herramientas, este protocolo no estará disponible y la forma de obtenerlo será por medio de la aplicación Vi Package Manager, esta aplicación se instala por defecto en nuestro ordenador cuando realizamos la instalación del software, por lo que únicamente tendremos que ir a Windows y buscar con el nombre “Vi Package Manager”


*Figura 192 Icono de la aplicación Vi Package Manager. Tomada de https://www.ni.com/es-co/support/downloads/tools-network/download.jki-vi-package-manager.html#443251*

Para continuar con la instalación, requerimos de estar conectados a internet ya que una vez abierta la aplicación, se comenzará a descargar información adicional de los paquetes disponibles, el tiempo de descarga y disponibilidad de la aplicación dependerá evidentemente, de la velocidad de conexión a internet con la que disponemos.

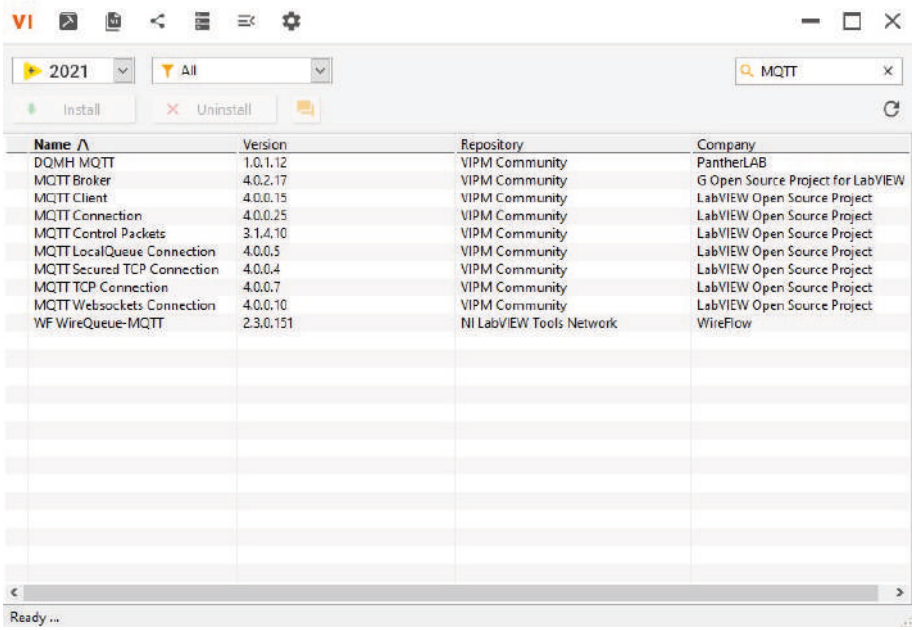



Figura 193 Interfaz Vi Package Manager. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Una vez en la interfaz, podemos dirigirnos directamente al campo de búsqueda donde digitaremos “MQTT” a lo que obtendremos los resultados evidenciados en la figura 192, para continuar con la instalación, seleccionaremos MQTT Client y automáticamente en la parte superior izquierda de la interfaz, se habilitará el botón de Install, el cual presionaremos

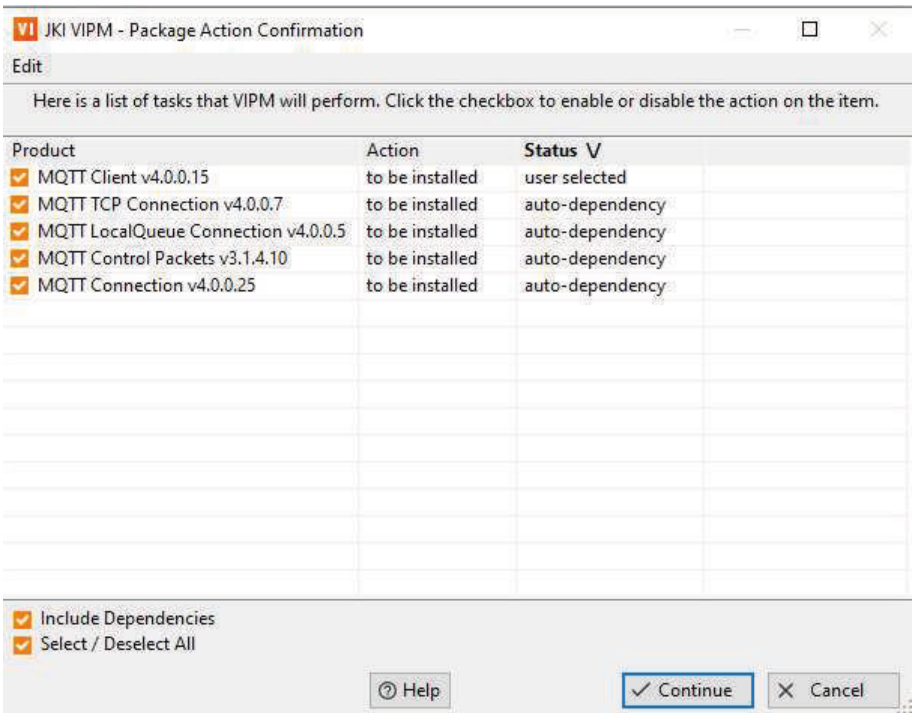



Figura 194 Productos Complementarios del MQTT. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Únicamente deberemos dar clic en Continue, inmediatamente comenzará la descarga del protocolo, los productos, herramientas y demás archivos complementarios, es importante resaltar que, una vez terminada la instalación, LabVIEW deberá reiniciarse, por lo que se recomienda guardar cualquier tipo de archivo o proyecto que se esté realizando para evitar pérdidas de información, posteriormente y como de costumbre, para toda instalación de productos adicionales se deberán aceptar los términos y condiciones definidos por el desarrollador del software, como se puede evidenciar en la siguiente Figura.

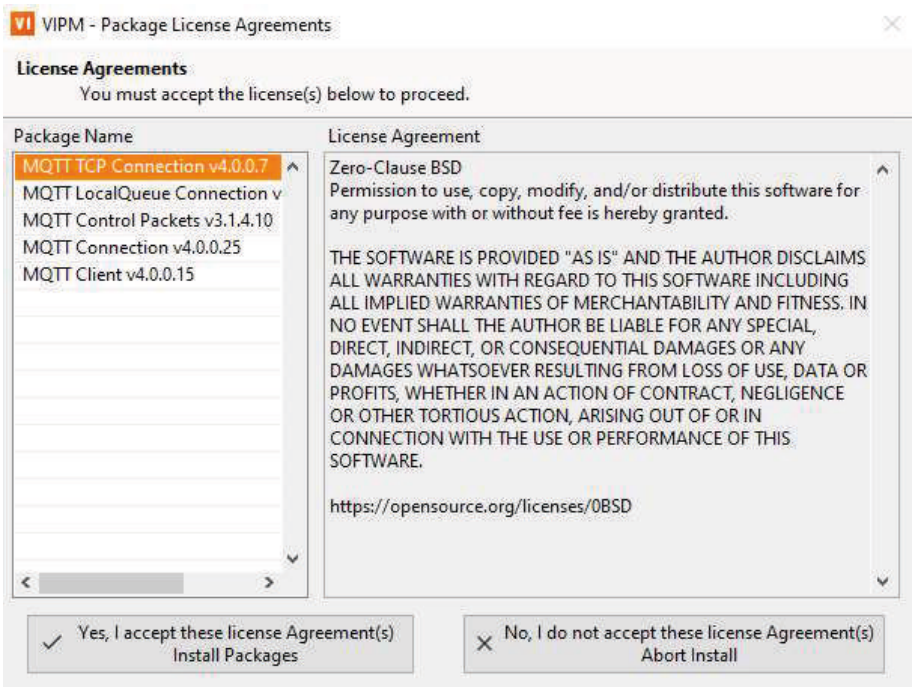



Figura 195 Ventana de Términos y Condiciones. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Una vez aceptados los términos, el software LabVIEW se reiniciará automáticamente y ahora, podremos evidenciar que disponemos de los bloques de MQTT para la realización de una conexión entre LabVIEW e IoT.

Ahora bien, para encontrar los protocolos inicialmente, debemos entrar a una serie de categorías y subcategorías, de tal forma que:

Abriendo el menú de funciones en el Diagrama de Bloques, en la categoría Addons > LabVIEW Open Source > MQTT, como se puede evidenciar en la siguiente figura.

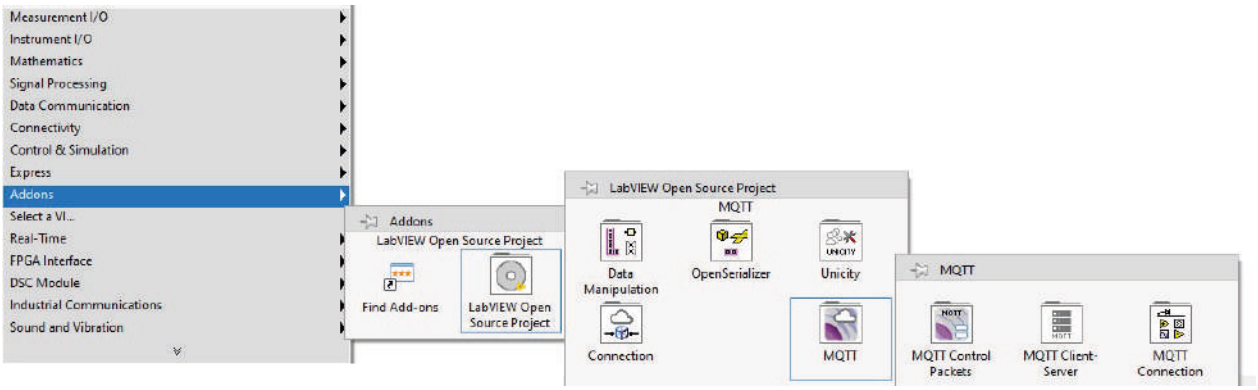



Figura 196 Categorías y Subcategorías MQTT. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Si entramos en MQTT Client – Server, encontraremos los bloques más sencillos para hacer una pequeña introducción al IoT, como mencionamos anteriormente, este capítulo tocará temas teóricos y superficiales, la investigación recaerá en el interés del estudiante y las posibilidades que determine para futuros proyectos.

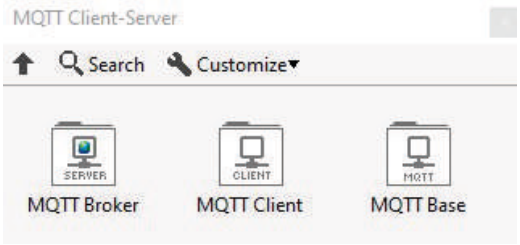
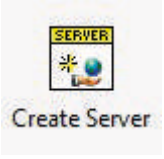
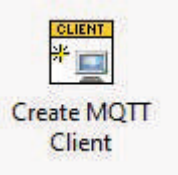



Figura 197 Categoría MQTT Client – Server. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*




Dentro de MQTT Broker, podemos encontrar nuestro bloque más importante, el cual es crear un servidor, como sabemos los servidores son los intermediarios entre nuestro modelo publicación – suscripción, donde nuestro servidor será nuestro MQTT Broker.




Dentro de MQTT Client, encontraremos nuestro cliente (publicación), el cual, dependiendo de nuestra aplicación o necesidad, podrá ser entendido como un sensor o algún tipo de herramienta que genere o “publique” un resultado que será enviado a nuestro MQTT Broker.


Por último, tendremos nuestro lector de MQTT (suscriptor) el cual se encargará de leer los datos arrojados por el intermediario y con él, podremos operar las variables a favor de nuestra aplicación.

Ahora bien, pondremos el siguiente ejemplo que tiene una característica muy importante, se evidenciará el ejercicio y su funcionamiento, el estudiante estará en la capacidad de determinar la lógica de programación con la cual, fue realizado.

El ejemplo se basa en un ejemplo sencillo, donde tendremos un publicador, un servidor y un suscriptor, el publicador se encargará de realizar el censo de la temperatura y enviar este valor medido al servidor, quien será el responsable de enviar el dato al suscriptor, quien procesará la información recibida y dará un resultado de acuerdo a la medición.

Para este ejemplo es necesario configurar tres aplicaciones en LabVIEW, se recomienda utilizar la herramienta de ayuda, para lograr comprender más sobre la importancia que tiene cada bloque dentro del aplicativo, estos bloques se colocan dependiendo de la aplicación, habrá proyectos donde no sea necesario, por ejemplo, generar una señal aleatoria entre 20 y 30, sin embargo, por comodidad y desarrollo del ejemplo, hay bloques adicionales que se prestan para el desarrollo del ejercicio.

Comenzaremos con la configuración e inicialización del servidor, esta configuración debe ser diseñada de la siguiente manera:

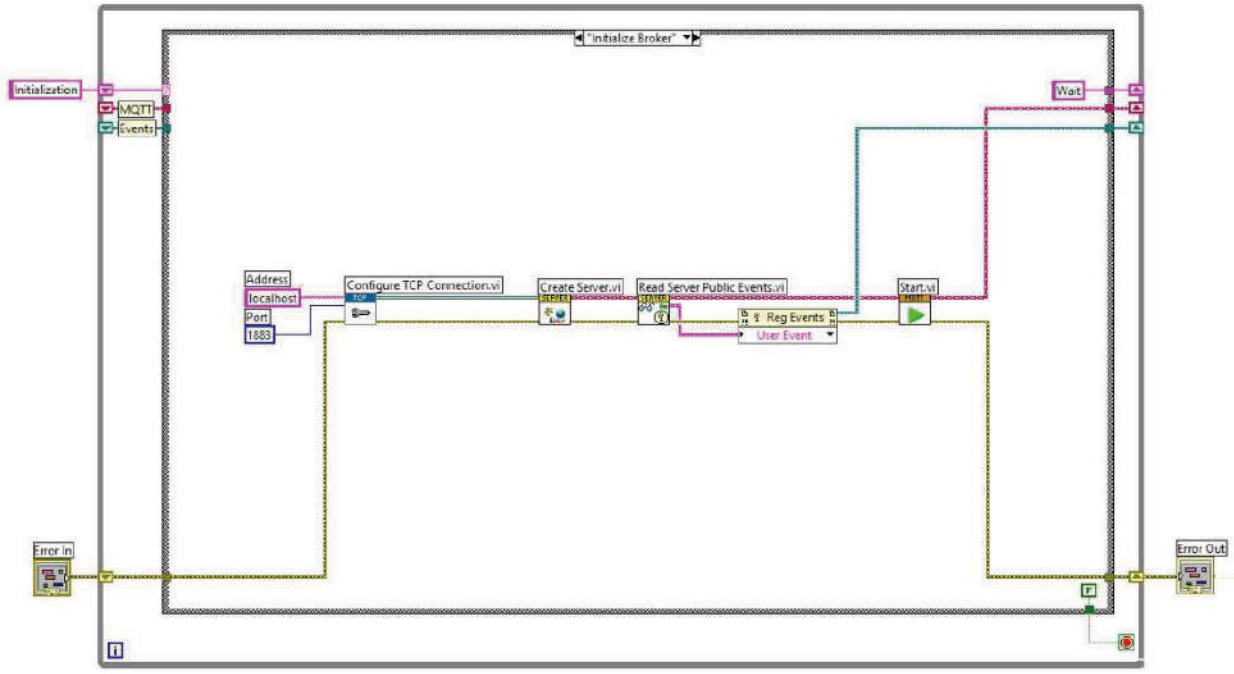



Figura 198 Inicialización Broker. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Nuestro Publisher, cuenta con una configuración basada en casos, donde tendremos que configurar los cuatro casos, de tal forma que:

- Inicializador del protocolo MQTT
- Tiempo de espera
- Datos de temperatura
- Publish Data (datos a enviar)

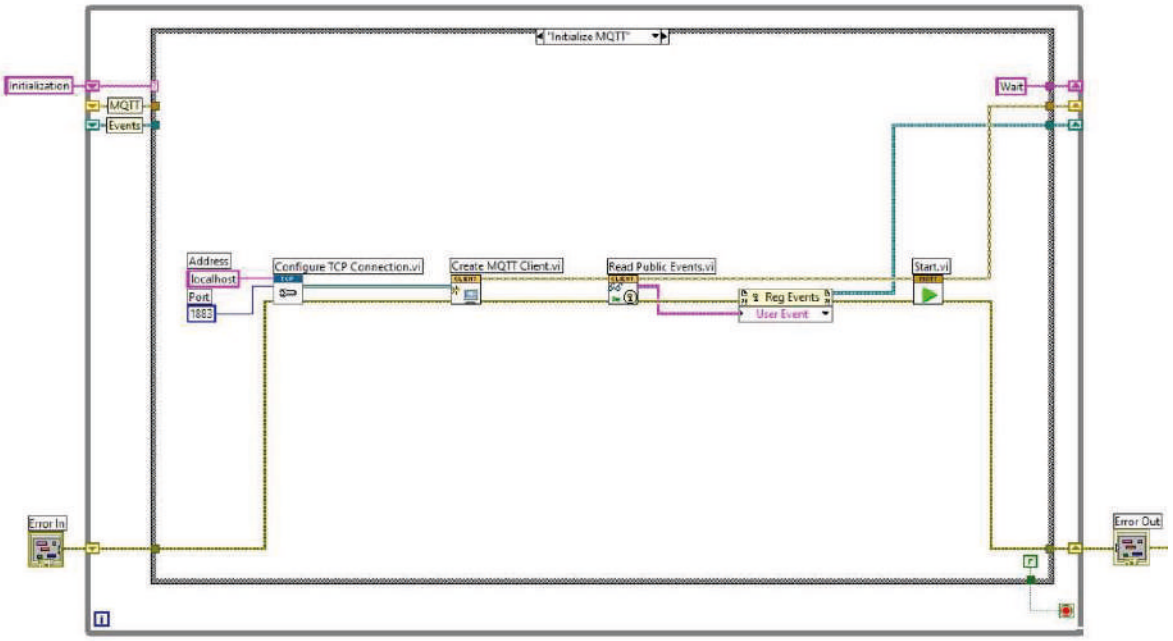



Figura 199 Caso Inicializar MQTT Publisher. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

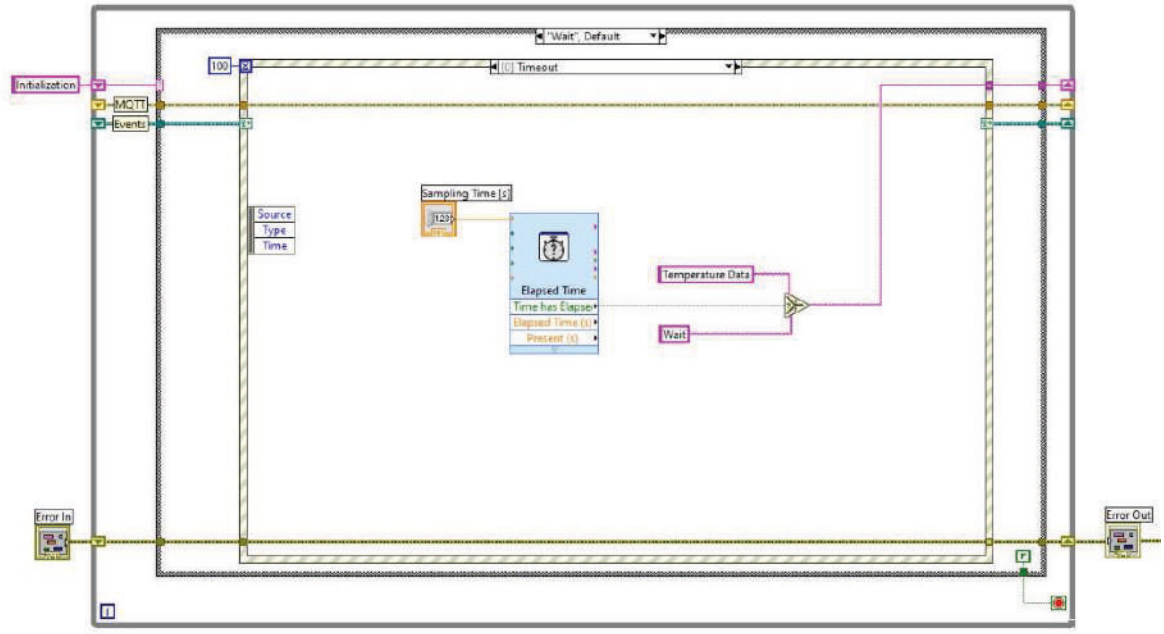



Figura 200 Caso Timer MQTT Publisher. Autoría Propia.

Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona

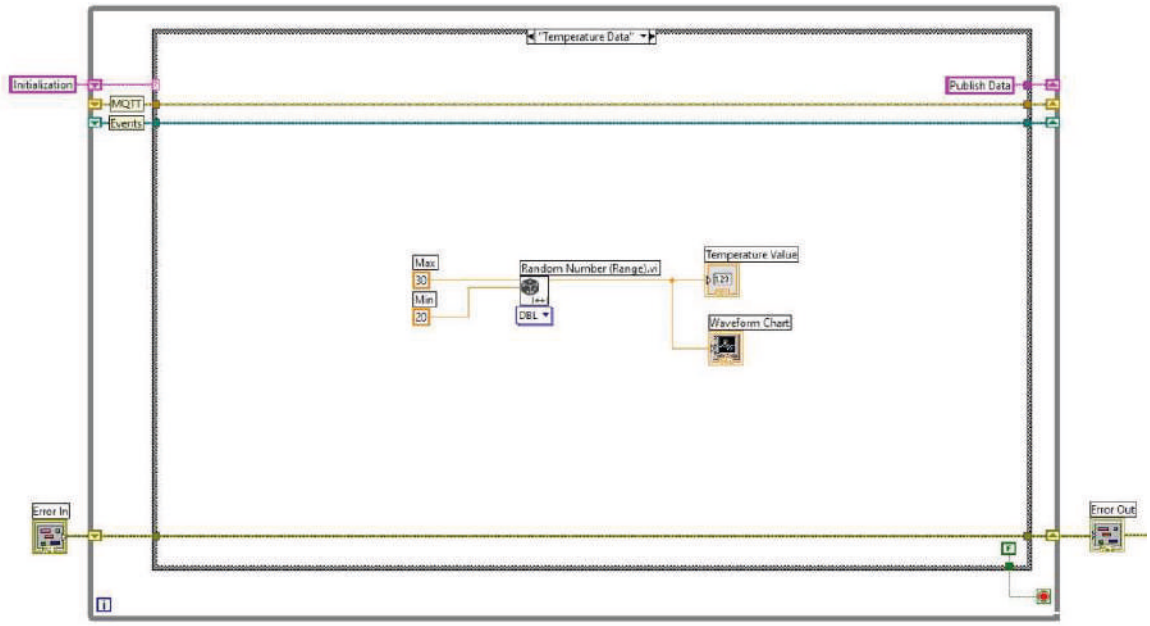



Figura 201 Caso Lectura "Aleatoria" de Temperatura. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

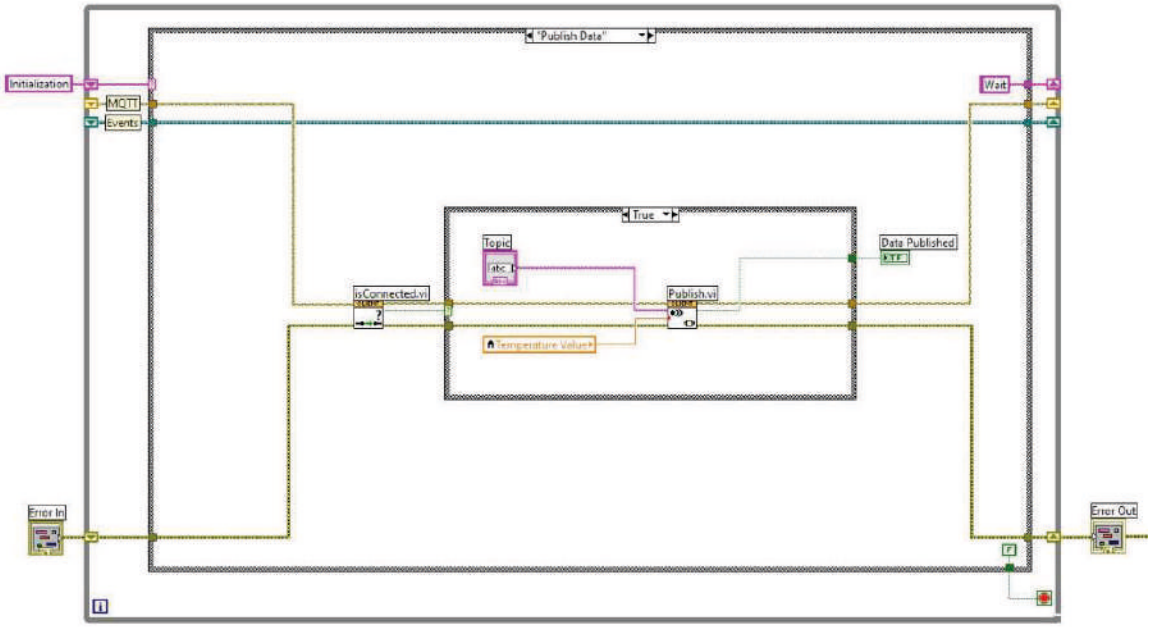



Figura 202 Caso Data Publisher. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Resumiendo, el caso Inicializador evidenciado en la figura 198 se encargará de iniciar el protocolo MQTT, la figura 199 corresponde al timer o delay con el que serán enviados los datos, la figura 200 contempla un simulador de señales de temperaturas por lo que obtendremos un valor entre 20 °C y 30 °C como lo muestra la figura, y, por último, el Publisher data, que será el encargado de enviar los datos al bróker.

Por último, inicializaremos y configuremos el subscriber, que se encargará de interpretar y mostrar los datos recibidos por el Broker, tendrá la siguiente configuración.

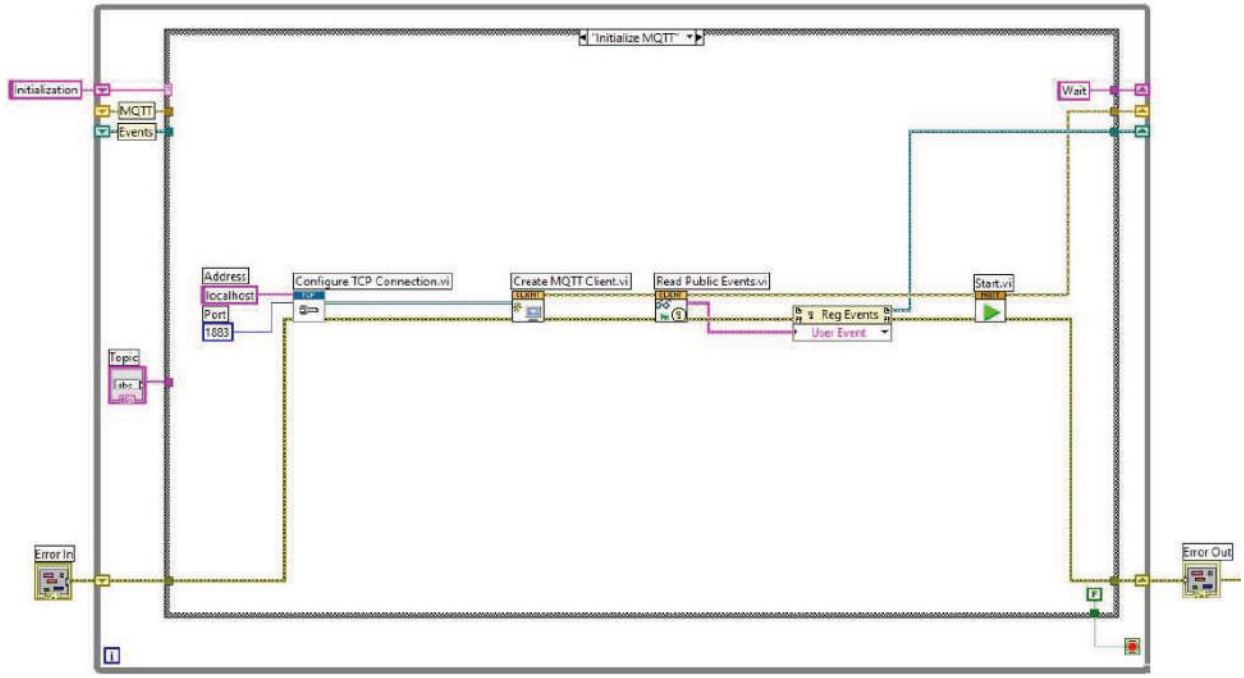



Figura 203 Inicializador MQTT Subscriber. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

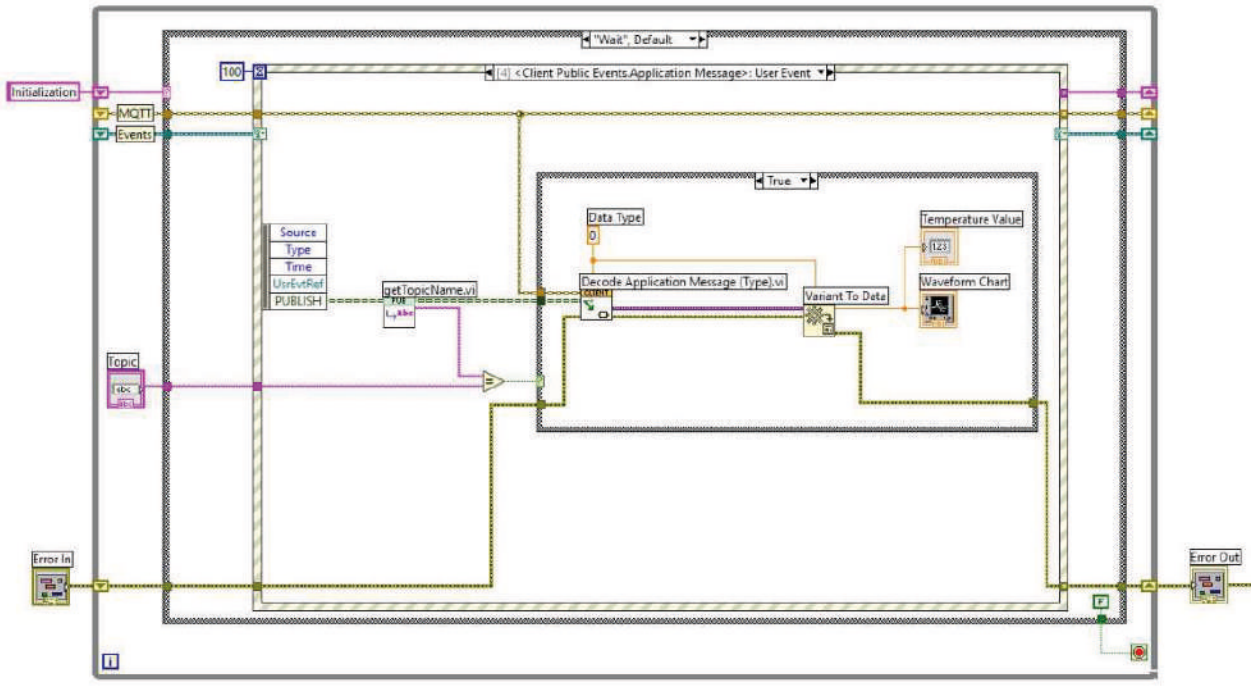



Figura 204 Grafica del valor recibido por el Broker. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ejecutando nuestros programas, obtendremos un sistema que simulado y aunque no parezca, puede representar claramente la ideología del IoT, puede no parecerlo en primera instancia debido a que todo se ejecuta en un mismo ordenador, sin embargo, la aplicación práctica se centra en el estudiante y su interés por la investigación y aplicación de sus conocimientos.

Simulando nuestro ejercicio, obtendremos los siguientes resultados, para cada uno (Publisher, Broker y Subscriber) la simulación estará determinada por las siguientes imágenes

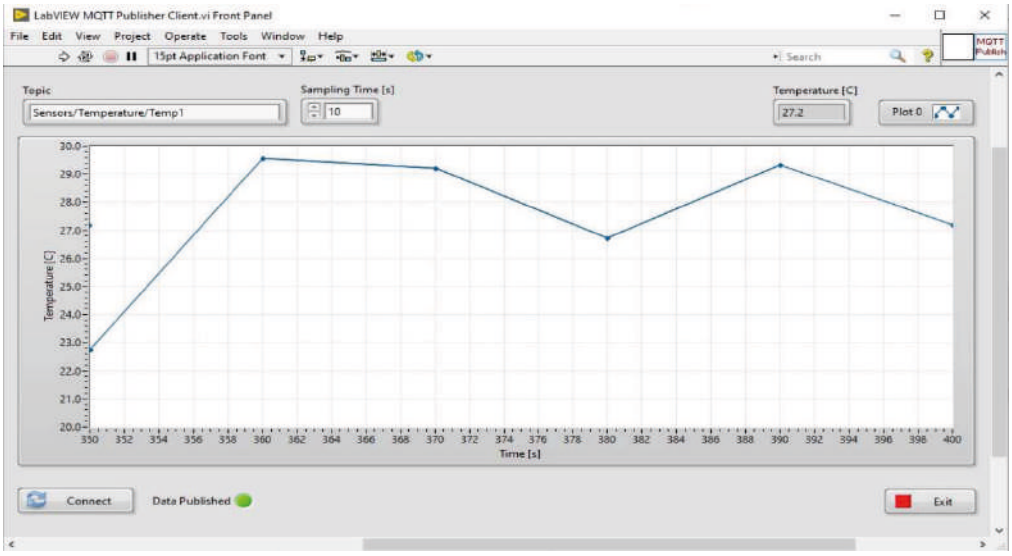



Figura 205 Señal Generada por el Publisher. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

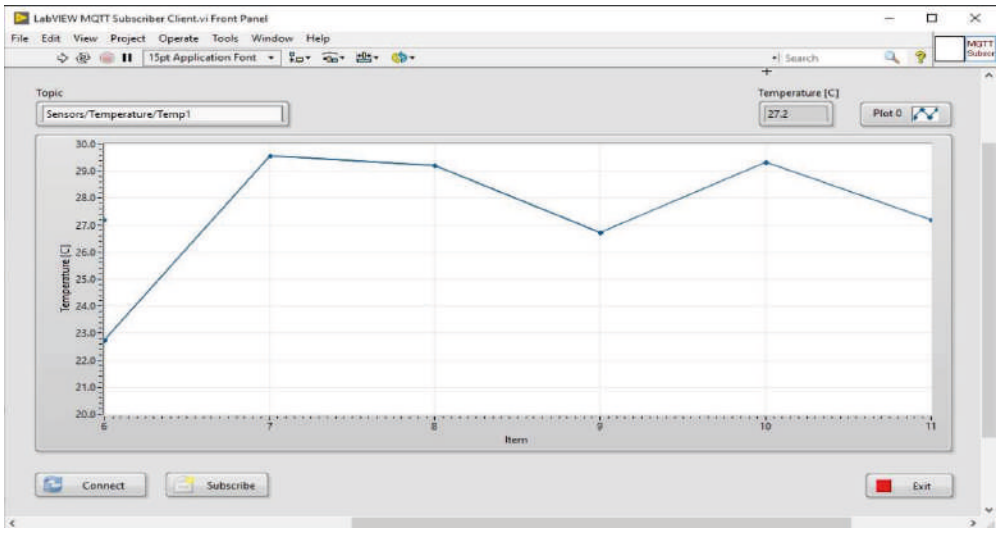



Figura 206 Señal Recibida por el Subscriber. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

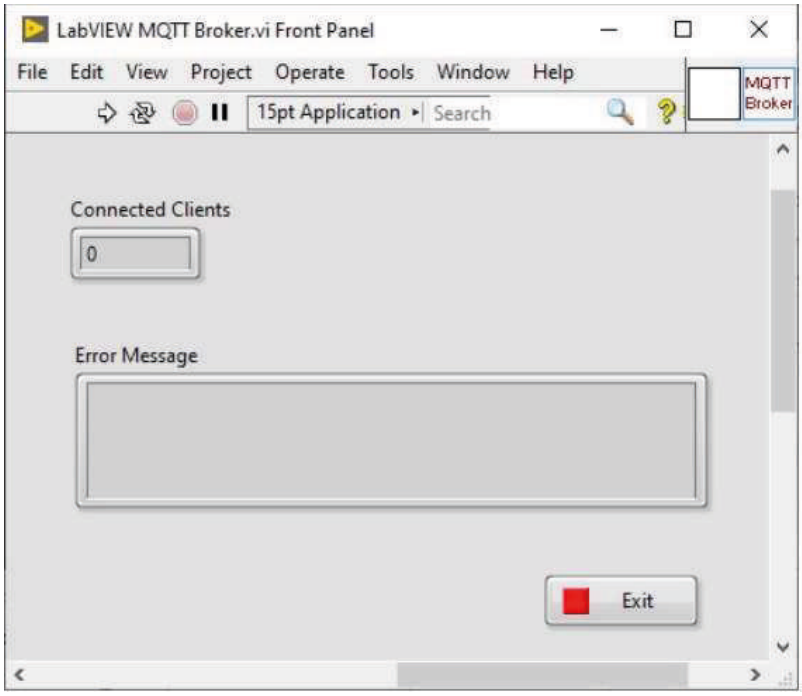



Figura 207 Broker o Intermediario. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como podemos apreciar en la figura 204 y la figura 205, la señal enviada y recibida es prácticamente la misma, lo importante es ver como la comunicación se produce por medio del Broker, quien bajo esta interfaz contiene la cantidad de clientes conectados durante el momento de recibir la señal y su respectivo Error.

### 4.2.2 **Posibles Aplicaciones**

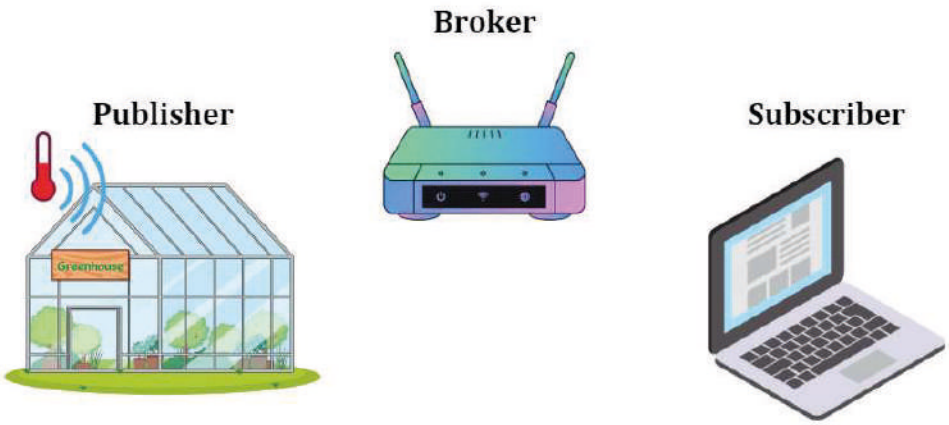



*Figura 208 Sensor para Temperatura de un Invernadero. Autoría Propia.*

IoT es un campo que tiene buena sinergia con el área de la agroindustria, una posible aplicación sería para evidenciar la temperatura de un invernadero, por lo que se pondría un sensor que estará midiendo la temperatura constantemente y enviando los datos al Broker, quien a su vez enviará los datos al Subscriber, este último puede estar ubicado en cualquier lugar, ya sea cerca del invernadero o a kilómetros de él, la importancia de IoT radica en que no necesariamente, la persona o el Subscriber, debe estar dentro del invernadero para poder evidenciar los datos.

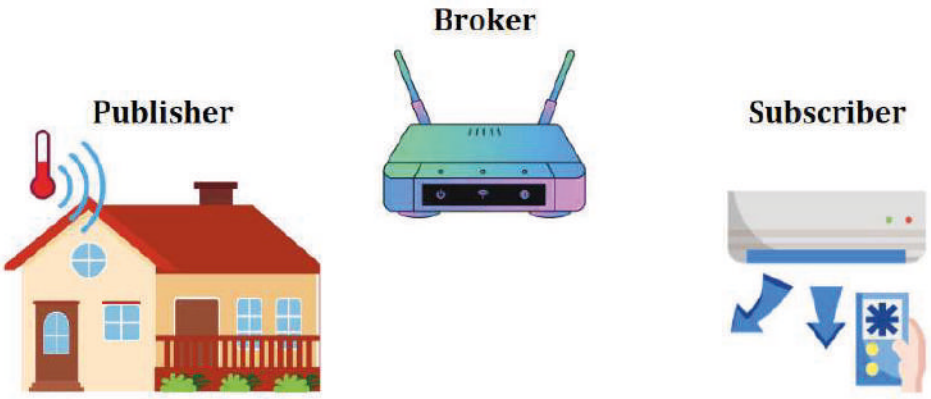



*Figura 209 Domótica con IoT. Autoría Propia.*

Otro ejercicio claro es el de Domótica, con este ejercicio práctico ponemos en aplicación dos campos muy importantes y de gran auge hoy en día, la Domótica y el IoT. El ejercicio puede ser tan sencillo como tomar la temperatura de una casa y posteriormente, con el Subscriber, analizar y controlar la temperatura, si la temperatura está por encima de la deseada, podemos enviar este dato a través del Broker a nuestro Subscriber, procesar la información y determinar si el aire acondicionado debe encenderse o no.

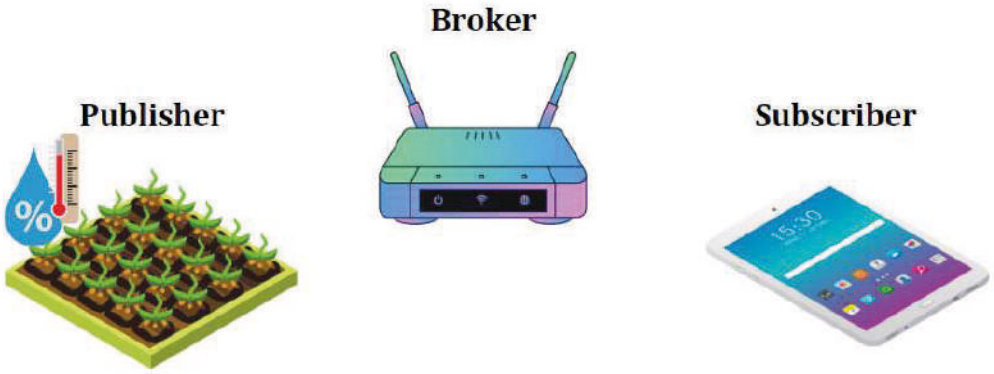



*Figura 210 Control Humedad con Tablet. Autoría Propia.*

Por último, recalcando la importancia de la agroindustria, proponemos un ejercicio similar, pero esta vez se busca medir la humedad, este porcentaje será enviado a

nuestro Broker quien posteriormente, lo enviará a la Tablet para el procesamiento de la información

## 4.3 AMQP


*Figura 211 Logo Protocolo AMQP.*

AMQP (Advanced Message Queuing Protocol) es un protocolo de comunicación de mensajería que se utiliza para el intercambio de mensajes entre aplicaciones. Es un estándar abierto y neutral desarrollado por la organización OASIS (Organization for the Advancement of Structured Information Standards)

Es importante resaltar que el protocolo es muy similar al visto anteriormente MQTT, ya que ambos tienen un funcionamiento similar, en modelo de publicación y suscripción unidos por un intermediario conocido como Broker, si lo comparamos con el protocolo MQTT evidenciamos una gran similitud

La principal diferencia varía en la cantidad de datos que manejar un protocolo en diferencia con el otro, el protocolo AMQP tiene un mejor procesamiento en flujos de datos mayores, por lo que la aplicación se ve mínimamente condicionada por el tipo de protocolo, siendo, para IoT, el protocolo MQTT el adecuado, por su facilidad y los pocos datos utilizados, para el ámbito empresarial, el protocolo AMQP es el recomendado

AMQP es más adecuado para aplicaciones empresariales y de integración de sistemas que requieren un modelo de comunicación más completo y opciones avanzadas de seguridad, mientras que MQTT es más adecuado para aplicaciones IoT que requieren un protocolo de mensajería más ligero y eficiente en términos de ancho de banda y uso de recursos.

### 4.3.1 **Instalar Protocolo AMQP**

Para la instalación de este protocolo, necesitaremos nuevamente la aplicación de Vi Package Manager, ya conocemos su icono y sabemos cómo ingresar, solo que únicamente buscaremos LabbitMQ, este complemento nos ayudará a conectarnos con un servidor RabbitMQ, el cual es un servidor basado en el protocolo AMQP, siendo eta la única opción para trabajar este protocolo dentro de LabVIEW.

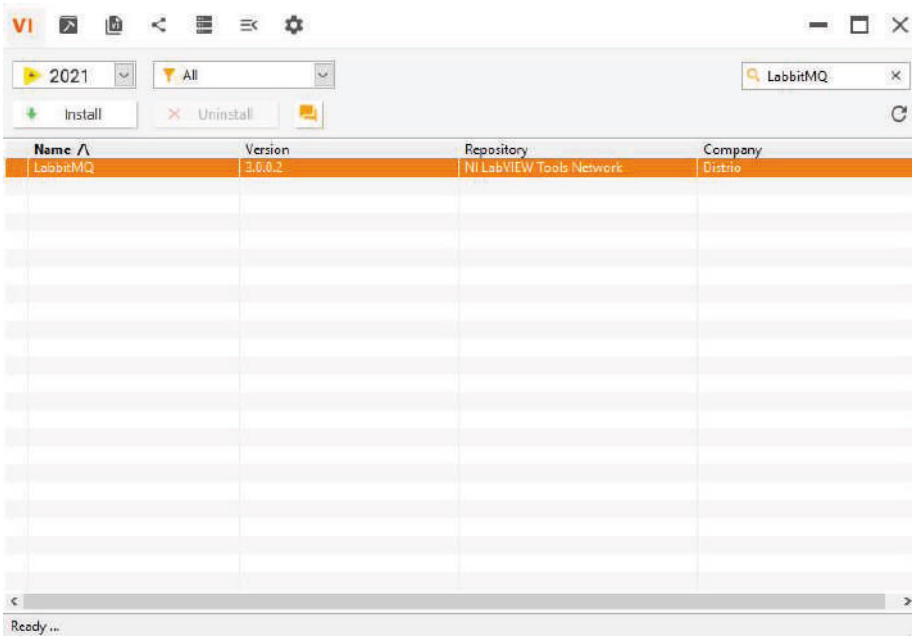



Figura 212 Búsqueda de Protocolo LabbitMQ. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Como evidencia la figura 211, solo encontraremos un complemento el cual, procederemos a instalar, posteriormente en una ventana adicional, saldrán los términos y condiciones los cual deberemos aceptar para poder continuar con la instalación, y, por último, nuestro LabVIEW se reiniciará por lo que se recomienda nuevamente, guardar los proyectos que estemos desarrollando.

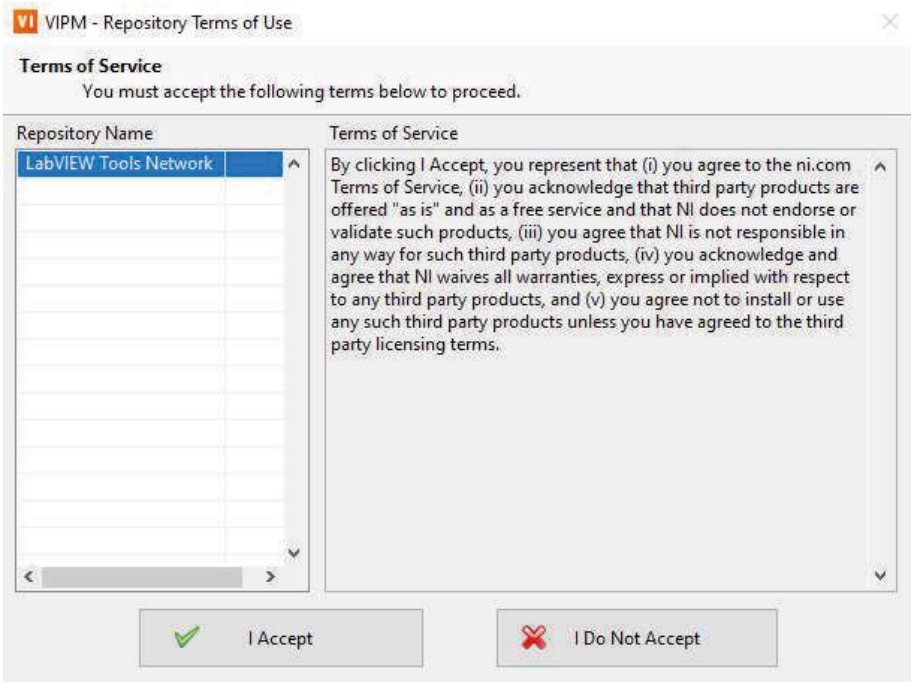



Figura 213 Términos de Servicio del Protocolo. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Debemos esperar la descarga del protocolo, que dependerá de la velocidad de internet que tengamos disponible.

Ahora bien, con el protocolo instalado, podemos crear un proyecto nuevo y en la ventana de Diagrama de Bloques, clicamos con el clic derecho en cualquier parte para poder acceder a las categorías que a tenemos instaladas, ahora buscamos la nueva categoría instalada.

Ingresaremos a la categoría Addons > Distrio > LabbitMQ

En esta subcategoría podemos acceder a todos los diagramas disponibles por este protocolo.

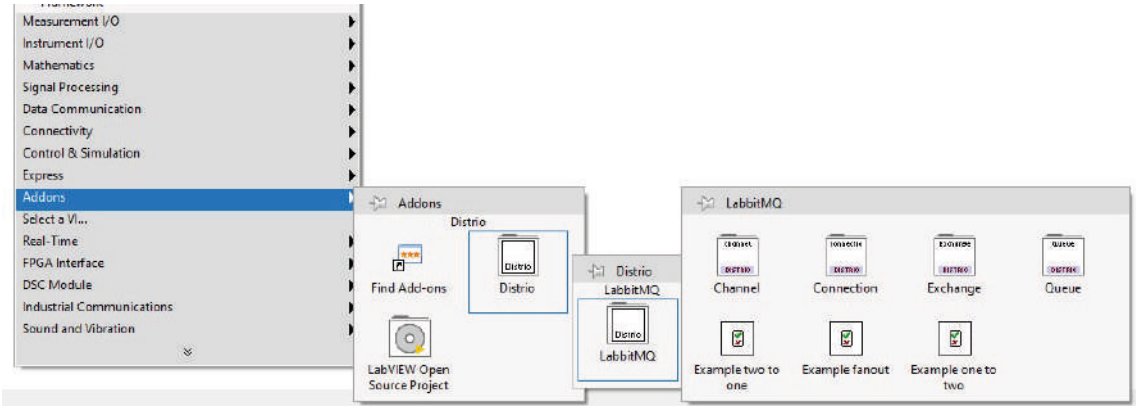



Figura 214 Categoría y subcategoría LabbitMQ. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*


En esta categoría, encontraremos los bloques necesarios para crear un ejercicio aplicación basado en el protocolo AMQP, estarán los diagramas de bloques que inicializarán el protocolo.


En esta categoría, los bloques están orientados a la conexión de los bloques, teniendo bloques que nos ayudarán a cerrar la conexión, crear un canal de conexión o crear una conexión directa.


En esta categoría, encontraremos tres bloques para el intercambio de información entre un bloque y otro, eligiendo mensajes predeterminados, propiedades de la conexión y demás información.


Dentro de Queue, encontraremos aquellos bloques que nos ayudarán a crear los receptores de o suscriptores de información, que serán de suma importancia para la creación adecuada de la comunicación basada en el protocolo AMQP

### 4.3.2 **Ejemplo Con Protocolo AMQP**

Si clicamos en este icono que se encuentra en la figura 213 se nos abrirá un archivo donde, estará programado ya un ejemplo, que básicamente funciona enviando, por medio de un Broker, un mensaje que aparecerá dentro de la misma interfaz siempre y cuando, la contraseña sea la correcta.

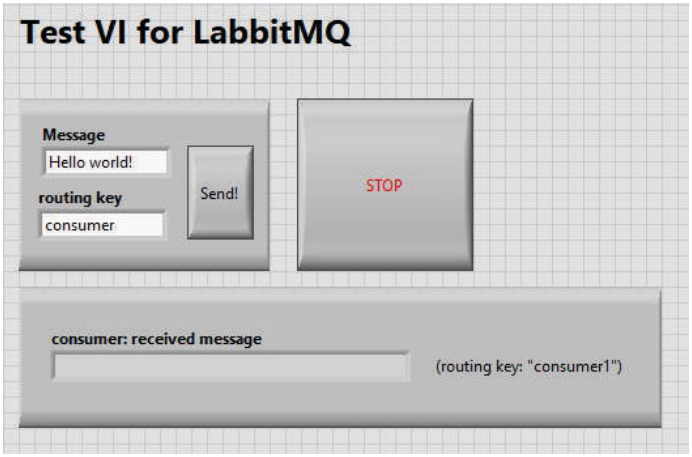



Figura 215 Interfaz Prueba LabbitMQ. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La interfaz puede apreciarse un poco sencilla, sin embargo, el código de programación tiene alto nivel de complejidad, por lo que tener claras las bases de los otros capítulos, es sumamente necesario para entender el siguiente código.

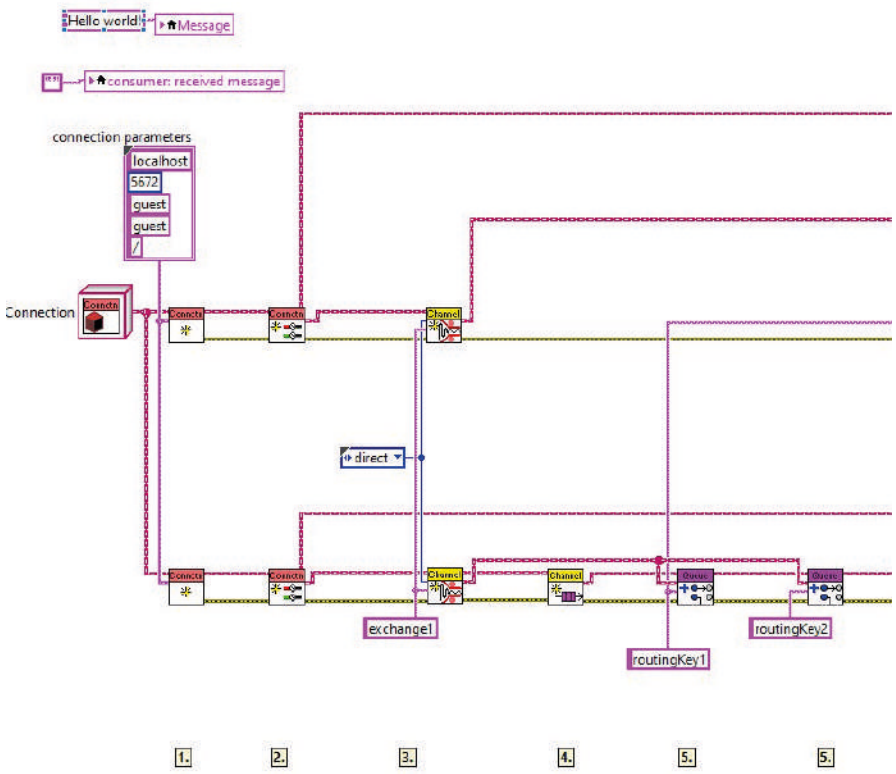



Figura 216 Ejemplo AMQP parte 1. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La primera parte, se centra en la conexión y la inicialización de los diagramas de bloque, como podemos apreciar el ejemplo está dividido en pasos, para que sea más sencillo asimilar la programación.

La segunda parte contempla un el código más detallado, usando el Broker y el lector del mensaje, el código es evidenciado a continuación en la figura 216.

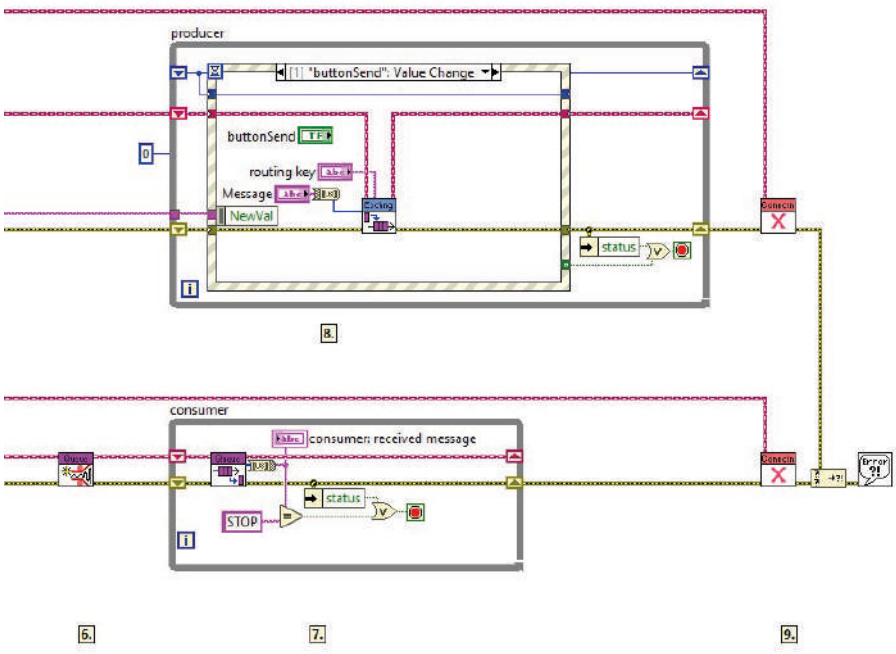



Figura 217 Ejemplo AMQP parte 2. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El código, puede ser explicado paso a paso, según la figura 215 y 216 de la siguiente forma:

1. Crear conexión con el servidor RabbitMQ

2. Crea un canal de comunicación en el servidor

3. Crea un intercambio privado

4. Registre una cola para recibir mensajes en

5. Agregar enlace con las claves de enrutamiento

6. Registrarse como consumidor (receptor)

7. Recibir mensajes en la cola que se envían a esta clave de enrutamiento

8. Enviar mensajes a un consumidor con la clave de enrutamiento

9. Cierra la conexión.

## 4.4 DDS


*Figura 218 Logo Protocolo DDS*

DDS (Data Distribution Service) es un protocolo de comunicación de datos en tiempo real que permite la distribución de datos y eventos entre diferentes sistemas y aplicaciones. DDS se basa en un modelo de publicación/suscripción, en el cual los productores de datos (o emisores) publican datos en un tema específico y los consumidores de datos (o receptores) se suscriben a ese tema para recibir los datos.

La similitud con los dos protocolos mencionados es evidente, tenemos nuevamente un Publisher, un Broker y un Subscriber, estos protocolos tienen mucho en común, la diferencia radica en el tipo de aplicación que tengamos por diseñar, por lo que elegir el protocolo adecuado será sumamente importante.

El DDS proporciona una arquitectura de middleware distribuido y escalable para la comunicación de datos en tiempo real, que es adecuada para sistemas de alta confiabilidad, como los sistemas críticos para la seguridad, la defensa y la industria. DDS es compatible con una amplia variedad de lenguajes de programación y plataformas, y proporciona un alto grado de flexibilidad y configurabilidad para adaptarse a diferentes requisitos de aplicaciones.

DDS ha sido adoptado en diversos sectores, como la industria aeroespacial, militar, automotriz y de telecomunicaciones, entre otros. También se ha utilizado en aplicaciones de IoT, como sistemas de monitoreo de salud, control de edificios inteligentes y automatización de fábricas.

### 4.4.1 **Instalar Protocolo DDS**

Para la instalación del Protocolo DDS volveremos a nuestro Vi Package Manager, por última vez durante el desarrollo de este libro, y procederemos a buscar RTI DDS en el buscador, seleccionamos el único protocolo y procedemos con la instalación.

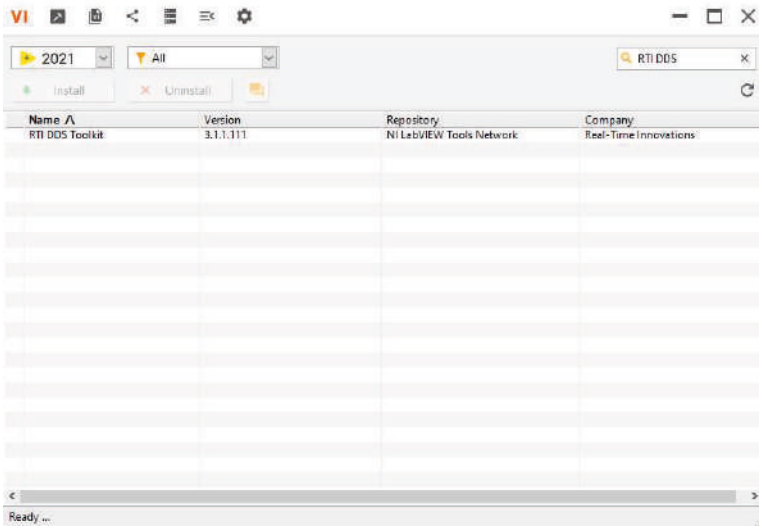



Figura 219 Búsqueda Protocolo DDS. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

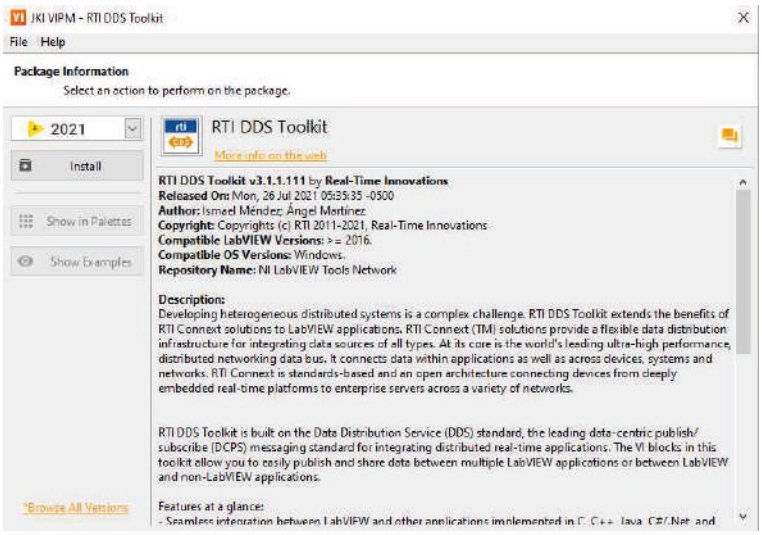



Figura 220 Descripción del Protocolo. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Clicamos en el botón de Install y comenzará la descarga, nuevamente la velocidad de instalación dependerá del internet, una vez completada la descarga y aceptados los términos, se reiniciará el programa y podremos acceder a los protocolos DDS en LabVIEW.

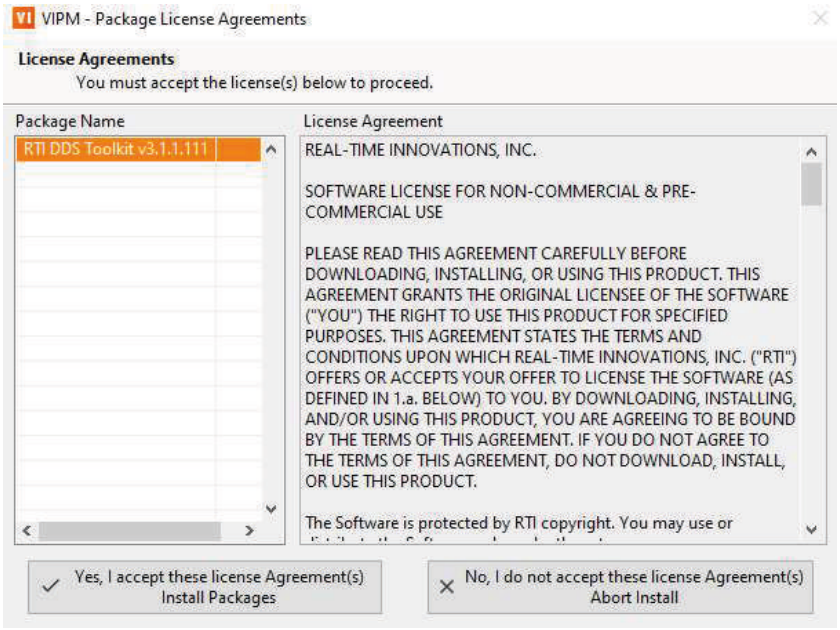



Figura 221 Aceptar Términos de Licencia. Autoría Propia.

*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

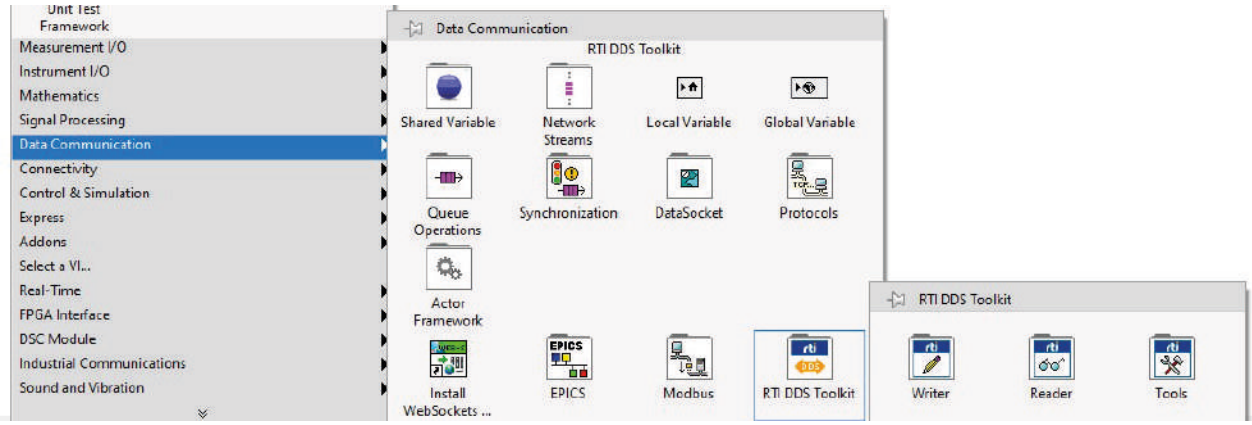
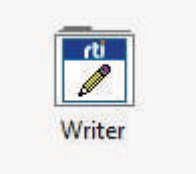



Figura 222 Categoría y Subcategoría para DDS. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

Ahora bien, para ingresar a las herramientas, debemos ingresar a Data Communication > RTI DDS Toolkit y entraremos al menú principal, donde tendremos tres categorías, Writer, Reader y Tools.

Nos enfocaremos inicialmente en Writer y Reader, partiendo de esto los Writer serán definidos como los Publisher o el dispositivo que enviará el dato, el Reader será el Subscriber que se encargará de leer lo que queramos enviar.




Este diagrama de bloque, nos ayudará a generar nuestro Publisher, el proyecto que enviará un dato hacia otro por medio del protocolo DDS, que más adelante, crearemos un aplicativo donde enviaremos una cadena de texto desde un archivo .vi a otro.


El lector, Reader o Subscriber, es el encargado de recibir la cadena te texto, aquí encontraremos diagramas de bloques que, dependiendo de la aplicación y el grado de complejidad se usará uno u otro bloque.

### 4.4.2 **Ejercicio Práctico**

Para este ejercicio, haremos una comunicación sencilla entre dos proyectos, para esto crearemos inicialmente nuestro Writer, donde ubicaremos una interfaz sencilla, únicamente para enviar una palabra, una cadena de caracteres o una frase, esto es un ejercicio netamente pedagógico.


Figura 223 Interfaz Gráfica Writer. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

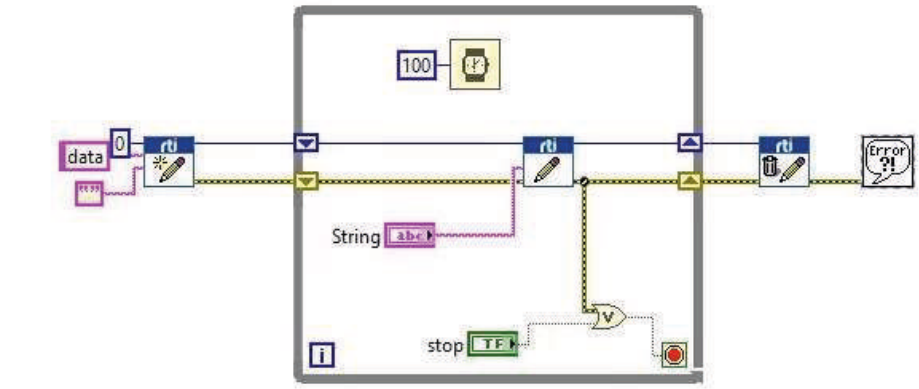



Figura 224 Diagrama de bloques del Writer. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

El código es un tanto sencillo, como vemos tenemos el Writer RTI que se encargará de enviar nuestro valor, claro está, que este código sirve para cualquier tipo de proyecto o aplicación, enfocada o no en IoT, partiendo de esto, con la creatividad del estudiante se pueden generar ejercicios o proyectos, a su vez, el docente puede utilizar este protocolo para crear nuevos ejercicios prácticos, donde podría comunicarse entre ordenadores dentro de la misma aula de clase.


Figura 225 Interfaz gráfica Receptor. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La figura 224, evidencia la interfaz gráfica del receptor, es similar a la del emisor con la diferencia de que de que nuestro campo String, no podrá ser editable y únicamente aparecerá el texto que sea enviado desde el receptor, esta comunicación es basada en el protocolo DDS, con la herramienta RTI DDS Toolkit.

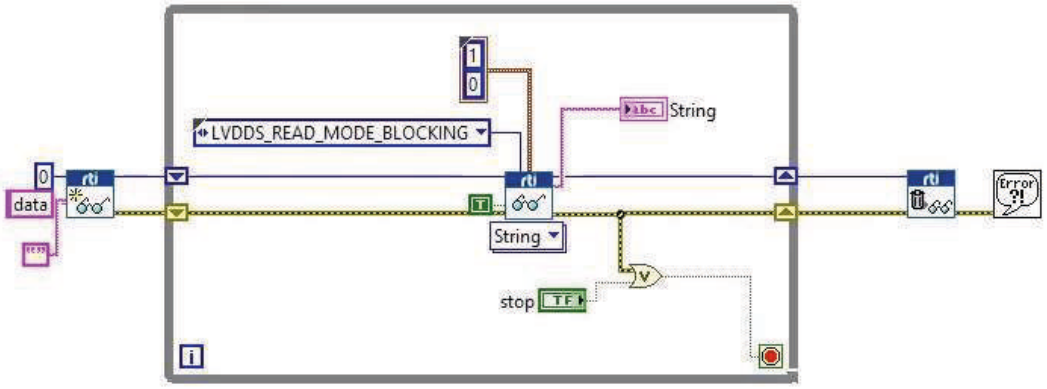



Figura 226 Diagrama de Bloques del Reader. Autoría Propia.
*Nota: El código de programación y las imágenes fueron usadas en el software LabVIEW bajo la licencia estudiantil de la Universidad de Pamplona*

La configuración de los bloques se realiza en modo recepción o lector, este diagrama funcionará como un lector, ya que tomará el valor que haya en el canal de comunicación y posteriormente, lo evidenciará en campo String evidenciado en la figura 225.

## APÉNDICE

Este libro, desarrollado en colaboración por los autores, pretende presentar una metodología o estrategia para todos los estudiantes que deseen tener una herramienta adicional durante el aprendizaje

La información presentada es recomendada para entender conceptos básicos de la programación en LabVIEW, como profesionales e investigadores recomendamos altamente ampliar estos conocimientos, buscar información adicional y complementaria que forme profesionales altamente competitivos.

Enfatizamos en la competitividad, ya que esta es una cualidad del ser humano que fomenta el progreso y la investigación, con esto pretendemos que el lector genere interés en todos estos temas que, en los últimos años, están teniendo un auge e importancia en nuestra sociedad actual.

Para los docentes, que desean usar esta herramienta como una metodología de aprendizaje, recomendamos la creación de ejercicios prácticos basados en la competitividad, la creatividad y la investigación.

# BIBLIOGRAFÍA


Alonso, R. (2020). *¿Qué es el Internet de las cosas (IoT) y por qué se le llama así?* HardZone. https://hardzone.es/reportajes/que-es/internet-cosas-iot/

Bañales, M. (2017). *Tipos de Datos*. Aprendiendo Arduino. https://aprendiendoarduino.wordpress.com/2017/10/14/tipos-de-datos-4/

Corredera, P. Á. (2022). *¿Qué es la programación basada en bloques?* CIBERNINJAS. https://ciberninjas.com/programacion-bloques/

Corsaro. (2010). *Servicio de Distribución de datos (Data Distribution Service), DDS*. Edu.Ec. https://www.utpl.edu.ec/proyectomiddleware/?q=tutorial-dds

Esteso, M. P. (2016). *Arduino y LabVIEW*. Geeky Theory. https://geekytheory.com/arduino-y-labview/

Interempresas. (2020). *LabVIEW, el software de ingeniería de sistemas que requieren pruebas, medidas y control*. Interempresas. https://www.interempresas.net/Electronica/Articulos/262150-LabVIEW-el-software-de-ingenieria-de-sistemas-que-requieren-pruebas-medidas-y-control.html

Lee, I., & Wallarm Inc. (2021). *What is AMQP Protocol ? All you need to know*. Wallarm.com. https://www.wallarm.com/what/what-is-amqp

Lucas, M. (2015). *Introducción a Arduino*. Edu.ar. http://cdr.ing.unlp.edu.ar/files/presentaciones/007_Introduccion%20a%20Arduino.pdf

Luis. (2019). *¿Qué es MQTT? Su importancia como protocolo IoT*. Luis Llamas; Luis. https://www.luisllamas.es/que-es-mqtt-su-importancia-como-protocolo-iot/

MasterHacks. (2013). *MANUAL BÁSICO DE PROGRAMACIÓN EN LABVIEW*.

Mecalux. (2014). *El trazo firme de LabVIEW*. Mecalux.es; Mecalux. https://www.mecalux.es/articulos-de-logistica/trazo-firme-labview

NI. (2017). *¿Qué es LabVIEW?* Www.ni.com. https://www.ni.com/es-co/shop/labview.html

Pascotto, A. (2014). *IMPLEMENTACIÓN DE UN SISTEMA DE CAPTACIÓN Y MONITORIZACIÓN, MEDIANTE INSTRUMENTACIÓN VIRTUAL, PARA LA MEDIDA DE DISTANCIA BASADO EN EL LVDT*. Universidad Politecnica de Valencia.

Rubio, M., & Javier, F. (2017). *Programa en Labview para el diagnóstico de la combustión en banco de ensayo*. Universidad de Valladolid.

Sahagun, S. (2021). *Aplicaciones industriales LabVIEW*. Blog Logicbus; Logicbus Blog. https://www.logicbus.com.mx/blog/aplicaciones-industriales-labview/

Sánchez, J., & Niño, E. (2017). *DISEÑO DE UNA INTERFAZ LABVIEW Y ARDUINO E IMPLEMENTACIÓN DE UN PROGRAMA APLICADO A LA MÁQUINA DE VACÍO DEL LABORATORIO DE MECÁNICA Design of an interface LabVIEW and Arduino and implementation of an applied program to the Máquina de vacío of Mechanical Laboratory Resumen*. Edu.co. Recuperado el 24 de marzo de 2023, de https://repository.udistrital.edu.co/bitstream/handle/11349/13449/S%C3%A1nchezHuertasJulietteXimena2018.pdf?sequence=1&isAllowed=y

Sánchez, O. (2010). *Desarrollo de una interfaz en ambiente de labview (Software) de control, super ol, supervisión y monit visión y monitoreo para el labor a el laboratorio de ingeniería orio de ingeniería eléctrica de la Universidad de La Salle*. Universidad de La Salle. https://ciencia.lasalle.edu.co/cgi/viewcontent.cgi?article=1539&context=ing_electrica

Sergio, C. (2018). *Introducción a Arduino*. Control Automático Educación. https://controlautomaticoeducacion.com/arduino/introduccion/


# SELLO EDITORIAL UNIPAMPLONA


## RESUMEN

LabVIEW,  un enfoque práctico para estudiantes, es un proyecto cuya idea nace a partir de la observación por parte de los autores, quienes evidenciaron dificultades en los estudiantes a la hora de aprender a programar dentro de este software, este recorrido por los temas básicos de LabVIEW será una gran herramienta para estudintes y educadores.



## CONTÁCTO

Km 1 Vía Bucaramanga Ciudad Universitaria
Pamplona - Norte de Santander
Teléfono: 607 5685303
315 3429495

atencionalciudadano@unipamplona.edu.co


ISBN: 978-628-7656-04-8