LABVIEW

Fase 4: Diseñar sistema de supervisión para el monitoreo de variables físicas

Presentado por

Dídier Javier Ramírez Henao

Presentado a

ALEXÁNDER FLÓREZ

Grupo

208055\_1

NOVIEMBRE 28 de 2021

Manizales – Caldas

**INTRODUCCIÓN**

La presente actividad consiste en la implementación de un sistema de supervisión con tarjetas de adquisición de datos para monitorear variables digitales y análogas de un proceso automatizado, por medio de comunicación serial. Debido a que el trabajo está enfocado en Arduino y LabVIEW, se encuentra en el documento información relacionada con estas plataformas.

Se tiene un componente individual que corresponde al desarrollo del tercer componente práctico del curso, en el cual, además de la interfaz gráfica en LabVIEW, se cuenta con la programación en Arduino y LabVIEW para lograr encender y apagar leds desde LabVIEW, activando pines digitales en Arduino y para leer una entrada análoga en Arduino, la cual es posible leer en LabVIEW. Todas estas variables se representan de forma gráfica.

En la actividad colaborativa se desarrolla un aplicativo que corresponde al subproceso de distribución, el cual además de las características del componente individual, muestra los niveles en que se encuentra el silo y activa el aspa de mezclado cuando el nivel es alto. La evidencia de la implementación de las tareas prácticas, además de estar sustentada en la programación realizada, se sustenta en los videos realizados, donde se comprueba que la actividad se ha desarrollado de forma satisfactoria.

**DESARROLLO DE LAS ACTIVIDADES**

1. **Consulte y documente lo siguiente:**
2. *¿Qué es una tarjeta Arduino, cuáles son las características de hardware?*

“Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino pueden leer entradas (luz en un sensor, un dedo en un botón o un mensaje de Twitter) y convertirlo en una salida, activando un motor, encendiendo un LED, publicando algo en línea. Puede decirle a su tablero qué hacer enviando un conjunto de instrucciones al microcontrolador en el tablero. Para hacerlo, utiliza el lenguaje de programación Arduino (basado en Wiring) y el Software Arduino (IDE), basado en Processing.

Todas las placas Arduino son completamente de código abierto, lo que permite a los usuarios construirlas de forma independiente y eventualmente adaptarlas a sus necesidades particulares. El software también es de código abierto”. (Xataka, 2021).

1. *¿Qué es el IDE de Arduino?*

“El IDE es un conjunto de herramientas de software que permiten a los programadores desarrollar y grabar todo el código necesario para hacer que nuestro Arduino funcione como queramos. El IDE de Arduino nos permite escribir, depurar, editar y grabar nuestro programa (llamados “sketches” en el mundo Arduino) de una manera sumamente sencilla”. (ARDUINO.cl, 2021).

1. *¿Qué aplicabilidad tiene la librería NI Visa de LabView?*

“NI-VISA es una API que proporciona una interfaz de programación para controlar instrumentos Ethernet/LXI, GPIB, seriales, USB, PXI y VXI en entornos de desarrollo de aplicaciones de NI como LabVIEW, LabVIEW NXG, LabWindows/CVI y Measurement Studio. La API se instala a través del controlador NI-VISA”. (Suplementario NI, 2021).

1. *¿Qué función cumple el software VI Package Manager?*

“El JKI VI Package Manager es un complemento de software que ayuda a encontrar e instalar complementos de LabVIEW desde la Red de Herramientas de LabVIEW. Se puede usar JKI VI Package Manager para crear bibliotecas de código reutilizable y herramientas para sus desarrolladores y clientes, y para organizar y dar mantenimiento a paquetes dentro de LabVIEW”. (NI JKI, 2021).

**COMPONENTE PRÁCTICO 3**

Figura 1. Interfaz gráfica\_Dídier Javier Ramírez Henao.

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

Enlace al video:

<https://drive.google.com/file/d/1ADPBgZ5YBZXxGdNX3Ze03JcSV6Yca6Ue/view?usp=sharing>

**SUBPROCESO DE MEZCLADO**

**Programación en LabVIEW:**

Figura 2. Ciclo while del subproceso.

Una captura de pantalla de una computadora

Descripción generada automáticamente

Interfaz de usuario gráfica, Diagrama

Descripción generada automáticamente

En la figura 2 se observa que en la parte externa del “while loop” se tienen dos registros. Un registro que tiene el propósito **de que al salir de dicho ciclo se conserve el estado al cual se debe dirigir el programa (inicio, fin, pausa, abortar, paro, nivel alto, nivel bajo o medio). El otro registro se llama NIVEL y allí se conserva el valor del nivel del silo. Se cuenta también con el bloque “VISA resource name”, el cual es para trabajar con la librería VISA y el bloque “VISA serial”, para trabajar con comunicación serial. Se tienen tres bloques de” VISA close” para cerrar la sesión de VISA de la parada, la marcha y la lectura del nivel de líquido.**

**Dentro del “while loop principal” se tiene:**

* **El stop, el cual para el programa y cumple la función de “abortar”.**
* **Dos selectores que vamos a denominar selector 1 y selector 2. El selector 1 depende del stop (abortar). En caso de que el Stop esté presionado, selecciona el estado “Abortar” y si no se presiona Stop, continúa el estado que viene por fuera del “while loop principal” a través del registro. El resultado de la selección va al selector 2.**
* **El selector 2 depende de varias variables booleanas (Marcha 3, inicio, NivelA, NivelM, NivelB); todas estas variables van unidas a través de compuertas OR, lo que quiere decir que, si al menos una de estas variables booleanas está activa, se toma la selección como verdadera y en caso de ser verdadera, se envía el estado que fue seleccionado a través del selector 1. Si ninguna de las variables booleanas está activa el selector 2 toma la opción de falso y toma el estado “Fin”. La decisión a través del selector se envía a un “case structure” donde según el estado se cumplen determinados procesos para el correcto funcionamiento de lo que se requiere del programa.**

**El bloque que está por fuera del “while loop”, denominado “VISA serial”, está conectado al bloque “Property Node”, en el cual van ingresando los bytes que llegan por el puerto serial desde Arduino y este boque está conectado al bloque “VISA read”, para leer la información que está en “Property Node”; después de leer esta información se usa el bloque “read buffer”, para extraer en formato string la información que hay en “VISA read”. El bloque “Fract/Exp String To Number” también está conectado a la salida del VISA read para convertir la información de string a números y el ofset se pone en cero para tomar la cadena de string desde el bit 0. Luego, la salida de “Fract/Exp String To Number” se conecta a la variable numérica denominada “Nivel” y en dicha variable se guarda el valor de la lectura análoga del potenciómetro que está conectado en Arduino, con un valor entre 0 y 1023.**

**El VISA serial también se encuentra conectado a dos VISA write, para que al pulsar en Labview los botones de parada y marcha, se activen los pines digitales de Arduino destinados a estas variables y se enciendan los leds de forma física y en caso de que no esté pulsada la marcha ni la parada, se apaguen los leds. Para ello, tanto en la marcha como en la parada se cuenta con un selector que dependen de la variable Marcha3 y Parada3 respectivamente. Si la marcha está activa se envía un 1 y si no lo está, se envía un 0 al VISA write; si la parada está activa se envía un 3 y si no lo está se envía un 2 al VISA write. Estos números llegan en formato string al VISA write y por medio de la comunicación serial y el código en Arduino, ponen en alto o en bajo los pines de parada y marcha.**

Figura 3. Estado INICIO.

Interfaz de usuario gráfica

Descripción generada automáticamente

**En el estado “INICIO” (figura 3), los indicadores booleanos se desactivan a excepción de los indicadores de marcha y parada, que dependen de los botones de Marcha3 y Parada3.**

**Dentro de este estado se encuentra un “case structure”, el cual funciona de la misma manera para los estados INICIO, NIVELA, NIVELM, NIVELB Y PARO así:**

**El case structure depende del botón de parada 3. Cuando se oprime el botón de parada 3, el case structure es verdadero y envía al estado de paro. Cuando el botón de paro no está activo, el case structure es falso y por lo tanto se compara el nivel del silo. Si el nivel del silo es bajo, se continúa al estado NIVELB, si es medio al estado NIVELM y si es alto al estado NIVELA. También se tiene una operación matemática para pasar el nivel del silo de la variable Nivel que está en un valor de 0 a 1023 a un valor entre 0 y 50 que es lo que se requiere. Para ello se multiplica la variable Nivel por la constante -0,048828125; a este valor se le añade 50 y con esto se tiene la variable NIVEL en un rango de 0 a 50. El resultado de la operación se actualiza en el registro para mostrarlo en la interfaz gráfica.**

Figura 4. Estado NIVELA.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**El estado NIVELA (figura 4), como su nombre lo indica, corresponde al NIVEL alto del contenido del silo. La programación es similar al estado de INICIO; las diferencias son: se tiene un tiempo de 100 milisegundos para activar y desactivar la variable booleana “AspaBoolean”, para simular el movimiento del aspa de mezclado cuando el nivel es alto. Se activan los indicadores booleanos NivelA y Mezclado para indicar el nivel alto del silo y que se está en el proceso de mezclado. Se ponen en falso los demás indicadores booleanos.**

Figura 5. Estado NIVELB.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**El estado NIVELB (figura 5), como su nombre lo indica, corresponde al NIVEL bajo del contenido del silo. La programación es similar al estado de INICIO; las diferencias son: Se activa el indicador booleano NivelB para indicar el nivel bajo del silo y se ponen en falso los demás indicadores booleanos.**

Figura 6. Estado NIVELM.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

**El estado NIVELM (figura 6), como su nombre lo indica, corresponde al NIVEL medio del contenido del silo. La programación es similar al estado de INICIO; las diferencias son: Se activa el indicador booleano NivelM para indicar el nivel medio del silo y se ponen en falso los demás indicadores booleanos.**

Figura 7. Estado PARO.

Captura de pantalla de computadora

Descripción generada automáticamente

**El estado PARO (figura 7), como su nombre lo indica, corresponde al paro del sistema, es decir, cuando se oprime el botón de parada3. La programación es similar al estado de INICIO; las diferencias son: Se ponen en falso todos los indicadores booleanos.**

Figura 8. Estado FIN.

Captura de pantalla de computadora

Descripción generada automáticamente

**En el estado FIN (figura 8), se sigue actualizando el nivel del contenido del silo. Los indicadores booleanos de marcha y parada se actualizan según el estado de los botones de Marcha 3 y Parada 3 y se envía al estado INICIO.**

Figura 9. Estado ABORTAR.

Captura de pantalla de computadora

Descripción generada automáticamente

**En el estado ABORTAR (figura 9), todos los indicadores booleanos se ponen en bajo y la variable NIVEL, que indica el nivel del silo se pone en cero.**

**Programación en Arduino:**

|  |  |
| --- | --- |
| **#include <SoftwareSerial.h>**  **int MarchaPIN = 13;**  **int ParadaPIN = 12;**  **int NivelPIN = 5;**  **int NivelValor;**  **void setup() {**  **Serial.begin(9600);**  **pinMode(MarchaPIN, OUTPUT);**  **pinMode(ParadaPIN, OUTPUT); El PIN asignado a la parada se define como salida**  **}**  **void loop() {**  **if(Serial.available())**  **{**  **int Datos = Serial.read();**  **if (Datos == '0')**  **{**  **digitalWrite(MarchaPIN, LOW);**  **}**  **if (Datos == '1')**  **{**  **digitalWrite(MarchaPIN, HIGH);**  **}**  **if (Datos == '2')**  **{**  **digitalWrite(ParadaPIN, LOW);**  **}**  **if (Datos == '3')**  **{**  **digitalWrite(ParadaPIN, HIGH);**  **}**    **}**  **NivelValor = analogRead(NivelPIN);**    **Serial.print((int)NivelValor); Serial.println(" Nivel");**    **delay(50);**  **}** | **Incluimos la librería SoftwareSerial**  **Se asigna el PIN 13 a la marcha del sistema**  **Se asigna el PIN 12 a la parada del sistema**  **Se asigna el PIN 5 al nivel del silo**  **Se define una variable entera para guardar el valor del nivel del silo.**  **Se indica la velocidad de la comunicación serial.**  **El PIN asignado a la marcha del sistema se define como salida.**  **El PIN asignado a la parada se define como salida.**  **Condición para validar si hay algún dato en el puerto serial que indique cambio de estado de la marcha o de la parada (desde LabVIEW).**  **Los datos de marcha o parada que llegan desde LabVIEW por comunicación serial se guardan en la variable Datos.**  **Si el dato que llega es un cero.**  **El pin de marcha se pone en bajo y no activa el led.**  **Si el dato que llega es un uno.**  **El pin de marcha se pone en alto y el led se enciende.**  **Si el dato que llega es el 2.**  **Es porque el botón de parada no está presionado y por lo tanto no se enciende el led de parada.**  **Si el dato que llega es un 3.**  **El pin de parada se pone en alto y se enciende el led.**  **Se lee el valor del nivel en la entrada análoga A5 y se guarda este valor en la variable NivelValor.**  **Se envía el valor del nivel por el puerto serial para ser mostrado en LabVIEW.** |

**Enlace a video subproceso de mezclado**

<https://youtu.be/FqfMJHo8pFI>

Figura 10. Se informa al grupo colaborativo el motivo de presentar el trabajo de forma individual.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

**CONCLUSIONES**

Existen gran diversas de sensores con diversidad de características que dependen de su función. Gracias a los sensores se pueden obtener señales eléctricas para lograr medir variables físicas, con el fin de determinar su comportamiento y dado el caso manipular dichas variables de acuerdo con las necesidades.

La librería VISA de LabVIEW permite establecer la comunicación serial con otros dispositivos y así leer y enviar datos, ofreciendo una interfaz gráfica amena e interactiva con los usuarios u operarios, por medio de la cual se pueden supervisar, controlar y manipular variables.

La mejor manera de implementar la solución fue primero verificar el funcionamiento de la programación del código de Arduino con el monitor serie, luego validar el funcionamiento con alguna aplicación bluetooth para Arduino que permita leer y enviar datos y finalmente probar el funcionamiento en LabVIEW.

Para un adecuado funcionamiento del prototipo diseñado es necesario conocer el funcionamiento de todos los componentes, el modo en que interactúan, la forma en que se deben programar y la forma en que deben ir conectados.

**REFERENCIAS BIBLIOGRÁFICAS**

¿Qué es Arduino, cómo funciona y qué puedes hacer con uno?. Xataka Basics. Recuperado de: <https://www.xataka.com/basics/que-arduino-como-funciona-que-puedes-hacer-uno>

Software de Arduino. ARDUINO.cl. Recuperado de:

<https://arduino.cl/programacion/>

Información General sobre NI-VISA. Visión general. Información suplementaria de NI. Recuperado de: <https://www.ni.com/es-co/support/documentation/supplemental/06/ni-visa-overview.html>

JKI VI Package Manager. NI Engineer Ambitiously. Recuperado de: <https://www.ni.com/es-co/shop/software/products/jki-vi-package-manager.html>

Yang, Y. (2014). LabVIEW Graphical Programming Cookbook. Packt Publishing. Páginas 155-161. Recuperado de: <https://bibliotecavirtual.unad.edu.co/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=690400&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_155>

Schwartz, M., & Manickum, O. (2015). Programming Arduino with LabView Páginas 7-20. Recuperado de <http://bibliotecavirtual.unad.edu.co/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=944047&lang=es&site=eds-live&ebv=EB&ppid=pp_7>

Ehsani, B. (2016). Data Acquisition Using LabVIEW. Packt Publishing. Páginas 27-30. Recuperado de <https://bibliotecavirtual.unad.edu.co/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=1440569&lang=es&site=eds-live&scope=site&ebv=EB&ppid=pp_27>

Flórez Martínez, A. (26,01,2017). Unidad 3\_Sistemas de adquisición de datos [Archivo de video]. Recuperado de: <http://hdl.handle.net/10596/10877>