DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNOLOGO EN ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN Y AVIÓNICA

TEMA: IMPLEMENTACIÓN DE UN INDUSTRUINO PARA PRÁCTICAS DE CONTROL DE PROCESOS EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS

AUTOR: ALEX XAVIER MALLITASIG MALLITASIG

DIRECTOR: ING. ZAHIRA PROAÑO C.

LATACUNGA

2017

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"IMPLEMENTACIÓN DE UN INDUSTRUINO PARA PRÁCTICAS DE CONTROL DE PROCESOS EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS"** realizado por el señor **MALLITASIG MALLITASIG ALEX XAVIER**, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor **MALLITASIG MALLITASIG ALEX XAVIER** para que lo sustente públicamente.

Latacunga, 09 de Febrero del 2017

ING. ZAHIRA PROAÑO C.

DIRECTOR

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

AUTORÍA DE RESPONSABILIDAD

Yo, **MALLITASIG MALLITASIG ALEX XAVIER,** con cédula de identidad N° 0503975062, declaro que este trabajo de titulación **"IMPLEMENTACIÓN DE UN INDUSTRUINO PARA PRÁCTICAS DE CONTROL DE PROCESOS EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS"** ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.

Consecuentemente declaro que este trabajo es de mi autoría en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 09 de Febrero del 2017

MALLITASIG MALLITASIG ALEX XAVIER

C.C. 0503975062

DEPARTAMENTO DE ELECTRÓNICA Y COMPUTACIÓN

CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

AUTORIZACIÓN

Yo, **MALLITASIG MALLITASIG ALEX XAVIER**, autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación **"IMPLEMENTACIÓN DE UN INDUSTRUINO PARA PRÁCTICAS DE CONTROL DE PROCESOS EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS"** cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.

Latacunga, 09 de Febrero del 2017

MALLITASIG MALLITASIG ALEX XAVIER

C.C. 0503975062

DEDICATORIA

Dedico este trabajo a las personas que creen que “el éxito en la vida no se mide por lo que logras, sino por los obstáculos que superas”.

AGRADECIMIENTO

Agradezco a mis padres y a mis hermanos que son un pilar importante en mi vida, que gracias a su apoyo y sus buenos consejos me han permitido culminar con éxito mi carrera profesional.

Agradezco a la universidad por darme la oportunidad de ser parte de una gran institución y por haberme permitido desarrollarme no solo en el ámbito profesional sino también en el ámbito personal.

Agradezco también a los docentes de la carrera por compartir sus conocimientos y consejos que han sido de gran importancia durante todo este tiempo.

ÍNDICE

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

El presente trabajo investigativo se realiza con la finalidad de representar un modelo de proceso de automatización común en la industria usando el Industruino I/O como controlador. Para esto se adquirirán las variables de nivel y temperatura para el llenado y vaciado del tanque cuyos transmisores se encuentran en el módulo didáctico PCT-3, donde se realizará lo siguiente, el fluido será suministrado desde un reservorio mediante una bomba centrífuga y en su trayecto al tanque pasará a través de un intercambiador de calor para bajar la temperatura del agua, una vez que el fluido alcance cierto nivel máximo en el tanque, se apagará la bomba y se encenderá el elemento calefactor que empezará a calentar el agua hasta alcanzar una temperatura máxima, una vez alcanzada la temperatura deseada se apagará el elemento calefactor y se activará la electroválvula a partir de este punto el proceso será cíclico. El Industruino I/O será programado utilizando el IDE de Arduino para que adquiera las señales analógicas de corriente y voltaje provenientes de los transmisores, mientras que la HMI será diseñada en LabVIEW donde se monitoreará el nivel y la temperatura del líquido, usando la comunicación serial entre el Industruino I/O y la PC. El controlador también ejecutará las instrucciones necesarias para interactuar con los elementos de control final en este caso la bomba, el elemento calefactor y la electroválvula.

PALABRAS CLAVE

- **TRANSMISOR**
- **NIVEL**
- **TEMPERATURA**
- **VOLTAJE**
- **CORRIENTE**

ABSTRACT

This research is done in order to represent a common automation process model in the industry by using the Industruino I/O as controller. It will be acquired variables of level and temperature to filling and draining the tank whose transmitters are in the didactic module PCT-3, in order to do these steps, the fluid will be supplied from a tank through a centrifugal pump to the tank will go through a heat exchanger to get down the temperature of water, once the fluid scope certain maximum level in the tank, the pump shuts off and the heater element will switch on, to heat the water until reach a maximum temperature, once reached the chosen temperature is switch off the element heater and is activated the solenoid valve from this point the process will stand cyclic. The Industruino I/O will be scheduled using the IDE of Arduino to acquire analogic signals of current and voltage from the transmitters, while the HMI will be designed in LabVIEW where it will monitor the level and the temperature of the liquid, using serial communication between the Industruino I/O and the PC. The driver also runs the necessary instructions to interact with the final control elements it means the pump, the heating element and the solenoid valve.

KEY WORDS

- **TRANSMITTER**
- **LEVEL**
- **TEMPERATURE**
- **VOLTAGE**
- **CURRENT.**

Lic. Wilson E. Villavicencio F. MSc.

DOCENTE UGT

CAPITULO I

1.1. Antecedentes

Debido a que el Industruino I/O es un equipo nuevo que se incorporó hace poco tiempo al mercado de los controladores lógicos programables, no existe información disponible de trabajos previos que se hayan realizado en la institución en base a este dispositivo.

1.2. Planteamiento del problema

El laboratorio de instrumentación virtual de la Unidad de Gestión de Tecnologías permite a los estudiantes de los últimos semestres de la carrera de electrónica realizar prácticas de laboratorio de las asignaturas acordes a su perfil de egreso, mismo que requiere incrementar sus equipos para desarrollar las prácticas, es así que la ausencia de un autómata programable versión Arduino, ocasiona que los conocimientos impartidos no sean acordes a las nuevas tecnologías.

De esta manera y por la gran importancia que involucra el desarrollo de prácticas de laboratorio con nuevos controladores programables, se ha planteado la implementación de un Industruino I/O, para prácticas de control de procesos en el laboratorio de instrumentación virtual, que permita a los estudiantes de electrónica desarrollar habilidades y destrezas, para reforzar el proceso de enseñanza mediante la práctica, aspecto de vital importancia para una educación de calidad.

1.3. Justificación e importancia

El trabajo se realiza con la finalidad de que el laboratorio de instrumentación virtual cuente con un controlador industrial versión Arduino, acorde a la tecnología actual, que ofrezca las prestaciones de un PLC, con la flexibilidad y sencillez de Arduino.

Con esto se consigue que los estudiantes tengan a disposición en el laboratorio un nuevo dispositivo electrónico para complementar la formación práctica con el objetivo de que la institución forme mejores profesionales y más competitivos mediante el uso de nuevas tecnologías.

1.4. Objetivos

1.4.1. General

Implementar un Industruino para prácticas de control de procesos en el laboratorio de instrumentación virtual de la Unidad de Gestión de Tecnologías.

1.4.2. Específicos

- Recopilar información sobre el Industruino I/O y sobre la programación del software para representar el modelo de un proceso industrial.
- Desarrollar el código de programación en el IDE de Arduino para adquirir las señales analógicas de los transmisores de nivel y temperatura del sistema físico PCT-3/2.
- Establecer una comunicación serial mediante un cable mirco USB para interconectar la PC y el Industruino I/O.
- Diseñar un HMI mediante el software LabVIEW para monitorear y controlar el proceso automatizado de llenado y vaciado de un tanque.

1.5. Alcance

El proceso trata de un líquido (agua) que será suministrado desde un reservorio mediante una bomba y en su trayecto al tanque pasa a través de un intercambiador de calor, una vez que el líquido alcanza cierto nivel máximo se apaga la bomba y se enciende el elemento calefactor el cual empieza a calentar el líquido hasta alcanzar una temperatura máxima, una vez alcanzada la temperatura deseada se apaga el elemento calefactor y se activa la electroválvula para drenar el líquido hacia el reservorio, cuando el nivel de líquido del tanque baja al nivel mínimo se cierra la electroválvula y se enciende nuevamente la bomba, a partir de este punto el proceso es cíclico. El proceso se llevará a cabo en el módulo didáctico PCT-3 que se encuentra en el laboratorio de instrumentación virtual de la Unidad de Gestión de Tecnologías.

La señal de 4-20mA del transmisor de nivel ultrasónico y la señal de 0-5V del transmisor de temperatura ingresarán al Industruino I/O para que sean procesadas y a continuación se ejecutarán las instrucciones programadas para que los canales de salida del Industrino realicen las acciones correspondientes a través de los elementos finales de control. El HMI será

desarrollado en LabVIEW, éste permitirá monitorear el proceso y realizar un paro de emergencia de ser necesario.

CAPITULO II

MARCO TEÓRICO

2.1. Controlador lógico programable (PLC)

Un PLC (Controlador Lógico Programable) es un equipo electrónico que opera de manera digital, almacena instrucciones para implementar funciones lógicas, secuenciales, temporizadas, de conteo y aritméticas en una memoria programable; para controlar mediante sus entradas y salidas máquinas y procesos. Por ello los PLC son utilizados en ambientes industriales donde se requieran tanto controles secuenciales como lógicos o ambos a la vez para actuar de manera rápida en la toma de decisiones y acciones para responder en tiempo real frente a los procesos. (Micro Capacitación, 2011)

2.1.1. Arquitectura interna de un PLC

2.1.1.1. Unidad central de Proceso

Está constituida en base a un microprocesador, se encarga de ejecutar el programa del usuario y producir las transferencias de datos desde las entradas hacia las salidas, además de encargarse de la comunicación con otros periféricos. Su función es tomar las instrucciones de la memoria, decodificarlas y posteriormente ejecutarlas. (Digital I - R.M.M., 2016)

2.1.1.2. Memoria del controlador

Los PLC cuentan con memorias con capacidad de almacenar información de manera permanente o provisional, es decir los PLC son capaces de almacenar datos del proceso como señales de entradas y salidas, variables internas, datos alfanuméricos y constantes también almacena datos de control como el programa del usuario y la configuración del PLC. Los controladores hacen uso de distintos tipos de memoria según sea su capacidad de almacenamiento, su velocidad de acceso, su volatilidad, etc. En un PLC la memoria interna es la que almacena el estado de las variables que maneja el controlador: entradas, salidas, temporizadores, señales de estado, etc. Esta memoria interna está caracterizada por la cantidad de bits que utiliza. (Digital I - R.M.M., 2016)

2.1.1.3 Interfaces de entrada/salida

Son las encargadas de establecer la comunicación con la planta, permiten ingresar la información proveniente de los sensores, interruptores, etc. (entradas) y enviar información a motores, bombas, electroválvulas y accionamientos en general. Para esto las interfaces deben filtrar, adaptar y codificar adecuadamente las señales. (Digital I - R.M.M., 2016)

2.1.1.4. Fuente de alimentación

La función de la fuente de alimentación en un controlador es suministrar la energía a la CPU y demás tarjetas según la configuración del PLC, + 5 V para alimentar a todas las tarjetas, + 5.2 V para alimentar al programador y + 24 V para los canales de lazo de corriente 20 mA. (Ramirez, 2005)

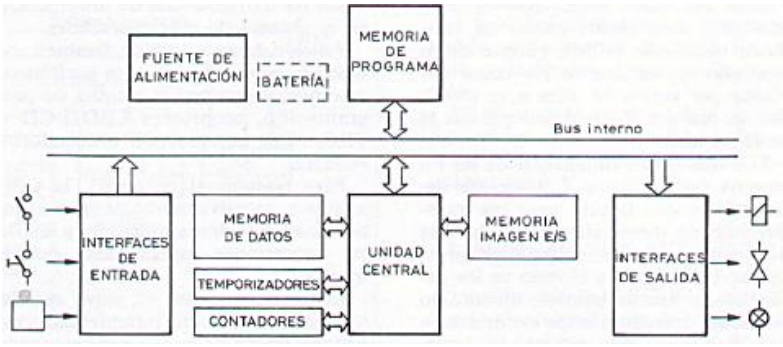


Figura 1 Estructura interna de un PLC

Fuente: (Digital I - R.M.M., 2016)

2.1.2. Funcionamiento

El PLC ejecuta en forma correlativa las instrucciones indicadas en el programa del usuario almacenado en su memoria. Entonces el PLC lee las entradas provenientes de la planta, ejecuta el programa con esos valores de entradas y genera las salidas (acciones) que controlan la planta. Esta secuencia se ejecuta continuamente para conseguir el control actualizado del proceso. (Digital I - R.M.M., 2016)

2.2. Industruino

De acuerdo con (Admin, 2015) Industruino es “un autómata programable que tiene como objetivo ofrecer una PLC potente con la flexibilidad y sencillez de Arduino”.

Figura 2 Industruino

Fuente: (Industruino, 2016)

2.2.1. Estructura modular de Industruino

2.2.1.1. Topboard

La Topboard es el cerebro e interfaz de usuario de Industruino, contiene un microcontrolador (Atmega32u4 o AT90USB1286) y la pantalla LCD. Todas las señales se conectan de esta tarjeta de conexión a la placa base mediante un cable de 40 pines FPC. (Industruino, 2016)

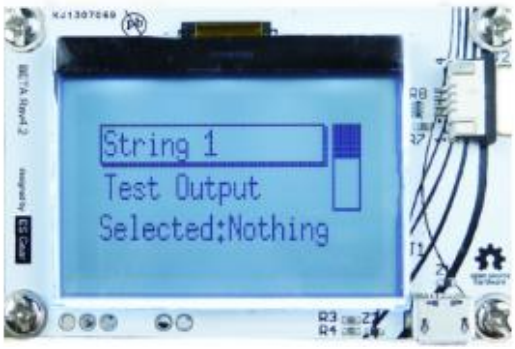


Figura 3 Topboard de Industruino

Fuente: (Industruino, 2016)

2.2.1.2. Versiones de la Topboard

a) Topboard 32u4

De acuerdo con (Industruino, 2016) la topboard 32u4 incorpora "un microcontrolador Atmega32u4 con 32 kB de flash, ideal para iniciar proyectos con Industruino".

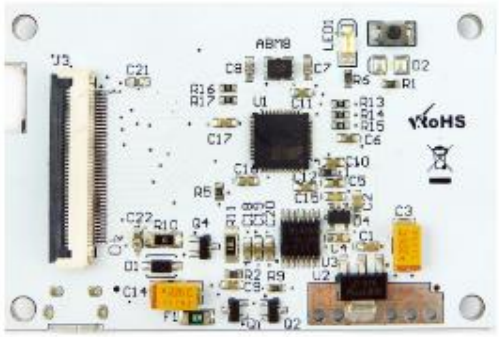


Figura 4 Topboard 32u4

Fuente: (Industruino, 2016)

b) Topboard 1286

De acuerdo con (Industruino, 2016) La topboard 1286 incorpora “el microcontrolador Atmega AT90USB1286 con 128kB de flash, para grandes bocetos de Arduino”.

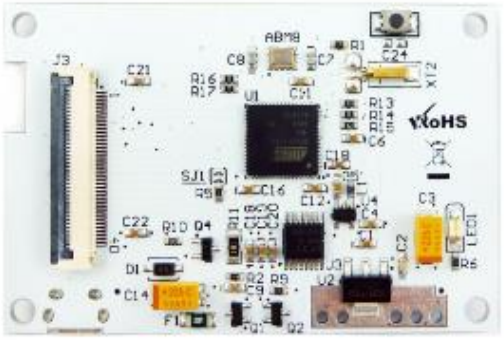


Figura 5 Topboard 1286

Fuente: (Industruino, 2016)

2.2.1.3. Placa base

De acuerdo con (Industruino, 2016) la placa base es “la placa de interfaz de entrada/salida de Industruino que permite la conexión con el mundo exterior mediante el uso conectores de tornillo. Actualmente Industruino ofrece dos tipos de placa base: PROTO e IND. I/O”.

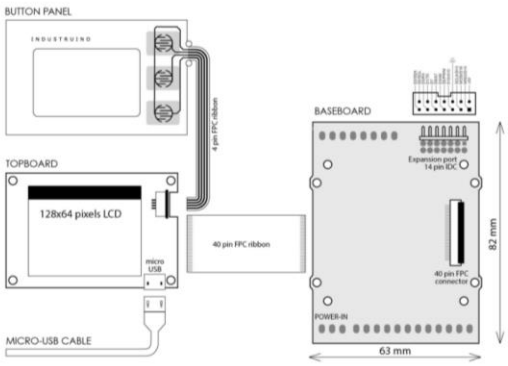


Figura 6 Placa base Industruino

Fuente: (Industruino, 2016)

2.2.1.4. Versiones de la placa base

a) Placa base PROTO

La placa base PROTO es sinónimo de "prototipos". El Industruino estándar viene con esta placa base para prototipos, el cual ofrece un gran espacio y flexibilidad para añadir sus propios componentes. Además posee un puerto de expansión de 14 pines IDC y un regulador de voltaje de 7-28Vin y >5V / 2A de salida. (Industruino, 2016)

Figura 7 Placa base IND. PROTO

Fuente: (Industruino, 2016)

b) Placa base IND. I/O

La placa base del IND. I/O es sinónimo de "nivel industrial de E/S". La tarjeta ofrece una abundancia de opciones de interfaz, y se ha aislado las fuentes de alimentación para cada una de sus tres zonas funcionales. Esta placa base es una solución de interfaz para cerrar la brecha entre la compatibilidad de Arduino y el mundo del PLC con sensores industriales y actuadores. Todos los periféricos de campo están aisladas desde el microcontrolador a través de aisladores digitales y se comunican a través del protocolo I2C. La placa incluye 8 canales 24V de I/O, 4 canales 0-10V / 4-20 mA ADC 18bit, 2 canales 0-10V/ 4-20 mA DAC 12 bits, transceptor RS485 aislado y zonas de potencia aislados. (Industruino, 2016)

Figura 8 Placa base del IND. I/O

Fuente: (Industruino, 2016)

2.3. Qué es Arduino?

De acuerdo con (Arduino, 2017) Arduino es “una plataforma electrónica de código abierto basado en hardware y software fácil de usar”.

2.3.1. Hardware de Arduino

Arduino está constituido en el hardware por un microcontrolador principal Atmel AVR, presente en la mayoría de los modelos de Arduino, encargado de realizar los procesos lógicos y matemáticos dentro de la placa, además de controlar y gestionar los recursos de cada uno de los componentes externos conectados a la misma. Además Arduino cuenta con la ventaja de tener entre sus elementos principales puertos seriales de entrada/ salida, lo que le permite conectarse por medio de un cable USB a una computadora para poder trabajar con ella desde el nivel software. (Weebly, S.f.)

2.3.1.1. Placa Arduino Leonardo

El Arduino Leonardo es una placa electrónica basada en el ATmega32u4. Cuenta con 20 pines digitales de entrada/salida (de los cuales 7 se pueden utilizar como salidas PWM y 12 entradas como analógicos), un oscilador de cristal de 16 MHz, una conexión micro USB, un conector de alimentación, una cabecera ICSP, y un botón de reinicio. Contiene todo lo necesario para apoyar el microcontrolador; basta con conectarlo a un ordenador con un cable USB o la corriente con un adaptador de CA a CC o una batería para empezar. (Arduino, 2017)

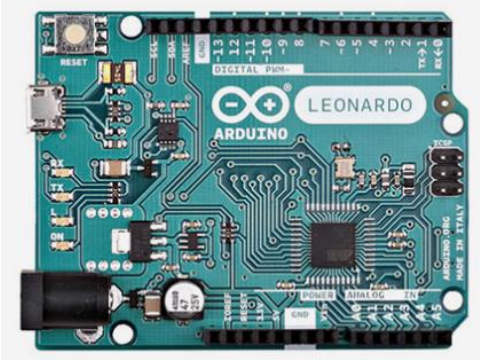


Figura 9 Placa Arduino Leonardo

Fuente: (Arduino, 2017)

2.4. Lenguaje de programación de Industruino

Para programar un Industruino se utiliza el IDE de Arduino, el cual se programa mediante el uso de un lenguaje propio basado en Processing, un lenguaje de programación de alto nivel que es similar a C++. Arduino

está basado en C y soporta todas las funciones del estándar C y algunas de C++. (Arduino, 2016).
Por lo tanto lo único que se necesita para controlar el Industruino I/O es la biblioteca Indio. Esta biblioteca aceptará la sintaxis de código que se utiliza en el IDE de Arduino y se hará cargo de toda la configuración y de la interfaz. (Industruino, 2016).

Figura 10 Software de Arduino

Fuente: (Arduino, 2016)

2.4.1. Librerías de Industruino

De acuerdo con (GitHub, Inc., 2016) La librería Indio utiliza "un expansor I2C para los canales de E / S por lo que también se necesita la librería Wire, estas son: #include <Indio.h> y #include <Wire.h>".

2.4.2. Estructuras del programa

De acuerdo con (Arduino, 2016) las estructuras de programa son: "setup () y Loop ()".

2.4.3. Estructuras de control

De acuerdo con (Arduino, 2016) las estructuras de control que se utilizan en el IDE de Arduino son: "if, if…else, for, switch case, while, break, return, goto".

2.4.4. Constantes

De acuerdo con (Arduino, 2016) las constantes que se utilizan en el IDE de Arduino son: "HIGH/LOW, INPUT/OUTPUT/INPUT_PULLUP, false, true".

2.4.5. Tipos de datos

De acuerdo con (Arduino, 2016) los tipos de datos que se utilizan en el IDE de Arduino son: “void, char, byte, int, word, float, string - char array, String – object”.

2.4.6. Entradas/salidas digitales

2.4.6.1. Configuración en modo entrada/salida

> Industruino I/O incorpora ocho canales de entradas/salidas digitales que pueden ser configurados mediante software, para esto basta colocar el número del canal que se va a utilizar y fijar el modo entrada (INPUT) o salida (OUTPUT). Para la configuración de las entradas/salidas digitales se utiliza “Indio.digitalMode(1,INPUT); // Fija CH1 como entrada, Indio.digitalMode(7,OUTPUT); // Fija CH7 como salida. (GitHub, Inc., 2016)

2.4.6.2. Leer/escribir

Para leer (Read) un valor digital únicamente seleccionar el canal deseado, mientras que para escribir (Write), seleccionar el canal y el estado que puede ser alto (HIGH) o bajo (LOW).

De acuerdo con (GitHub, Inc., 2016) para leer/escribir en las entradas/salidas digitales se utiliza “Indio.digitalRead(1); // Lee CH1, Indio.digitalWrite(7,LOW); // Fija CH7 en bajo (0V)”.

2.4.7. Entradas analógicas

> La sección analógica de E / S está aislada galvánicamente de la sección digital de E / S y de la sección del microcontrolador, para permitir una alimentación separada de la sección analógica obteniendo así una precisión óptima. En caso de que los sensores/actuadores analógicos estén en la misma fuente de alimentación con la sección digital (Vin 12/24V), tiene que conectar el GND analógica a la GND digital. (GitHub, Inc., 2016)

2.4.7.1 Configuración de la resolución del ADC

La configuración de la resolución de las entradas analógicas se puede fijar en valores que acepta el convertidor analógico digital (ADC) es decir valores de 12, 14, 16 y 18 bits.

De acuerdo con (GitHub, Inc., 2016) para configurar la resolución del ADC se utiliza “Indio.setADCResolution(16); // Fija la resolución del ADC

seleccione: 12bit@240SPS, 14bit@60SPS, 16bit@15SPS and 18bit@3.75SPS".

2.4.7.2. Configuración del modo de entrada

De acuerdo con (GitHub, Inc., 2016) Industruino I/O incorpora "cuatro canales de entradas analógicas, que pueden ser configurados mediante software, para esto basta colocar el número del canal que se va a utilizar y fijar el modo ya sea voltaje (0-10V) o corriente (4-20mA)".

a) Entrada de voltaje

Para configurar un canal como entrada analógica de voltaje, colocar el número de canal que se va a utilizar seguido de una de las siguientes opciones de entrada: modo V10 (0-10V), modo V10_p (0-100%), modo V10_raw (0-4096). Para configurar el modo de entrada en modo de voltaje se utilizan las siguientes opciones: Indio.analogReadMode(1, V10); //Fija la entrada analógica CH1 a 10V en modo (0-10V). Indio.analogReadMode(1, V10_p); // Fija la entrada analógica CH1 a porcentaje (%) en modo (0-10V -> 0-100%). Indio.analogReadMode(1, V10_raw); // Fija la entrada analógica CH1en modo 10V y lee el valor del ADC (0-10V -> 0-4096) (GitHub, Inc., 2016)

b) Entrada de corriente

Para configurar un canal como entrada analógica de corriente, colocar el número de canal que se va a utilizar seguido de una de las siguientes opciones para el modo de entrada: modo mA (4-20mA), modo mA_p (0-100%), modo mA_raw (0-4096). Para configurar el modo de entrada en modo de corriente se utilizan las siguientes opciones: Indio.analogReadMode(1, mA); //Fija la entrada analógica CH1 en modo mA (4-20mA). Indio.analogReadMode(1, mA_p); // Fija la entrada analógica CH1 en modo porcentaje en mA (4-20mA -> 0-100%). Indio.analogReadMode(1, mA_raw); // Fija la entrada analógica CH1 en modo mA y lee el valor del ADC (4-20mA -> 0-4096). (GitHub, Inc., 2016)

2.4.7.3. Leer una entrada analógica

Para leer (Read) un valor analógico únicamente se coloca el número correspondiente al canal que se va a utilizar y la lectura dependerá del modo de seleccionado.

De acuerdo con (GitHub, Inc., 2016) para leer una entrada analógica se utiliza "Indio.analogRead(1); // Lee la entrada analógica CH1 (la salida depende del modo seleccionado).

2.4.8. Salidas analógicas

2.4.8.1. Configuración del modo de salida

Industruino I/O incorpora dos canales de salidas analógicas, donde el modo de la salida puede ser establecido mediante software, para esto basta colocar el número del canal que se va a utilizar y fijar el modo ya sea voltaje (0-10V) o corriente (4-20mA).

a) Salida de voltaje

Para configurar un canal como salida analógica de voltaje, colocar el número de canal que se va a utilizar seguido de una de las siguientes opciones de salida: modo V10 (0-10V), modo V10_p (0-100%), modo V10_raw (0-4096). Para configurar la salida analógica en modo de salida de voltaje se utilizan las siguientes opciones: Indio.analogWriteMode(1, V10); // Fija la salida analógica CH1 a 10V en modo (0-10V). Indio.analogWriteMode(1, V10_p); // Fija la salida analógica CH1 a porcentaje de 10V en modo (0-100% -> 0-10V). Indio.analogWriteMode(1, V10_raw); // Fija la salida analógica CH1 en modo 10V and toma los valores del rango del DAC (0-4096 -> 0-10V). (GitHub, Inc., 2016)

b) Salida de corriente

Para configurar un canal como salida analógica de corriente, colocar el número de canal que se va a utilizar seguido de una de las siguientes opciones de salida: modo mA (4-20mA), modo mA_p (0-100%), modo mA_raw (0-4096). Para configurar la salida analógica en modo de salida de corriente se utlizan las siguientes opciones: Indio.analogWriteMode(1, mA); // Fija la salida analógica CH1 a modo mA (0-20mA). Indio.analogWriteMode(1, mA_p); // Fija la salida analógica CH1 a mA en modo porcentaje (0-100% -> 4-20mA). Indio.analogWriteMode(1, mA_raw); // Fija la salida analógica CH1 a modo mA and toma los valores en el rango del DAC (0-4096 -> 0-20mA). (GitHub, Inc., 2016)

2.4.8.2. Escribir en una salida analógica

Para escribir (Write) un valor analógico únicamente se coloca el número correspondiente al canal que se va a utilizar y la salida dependerá del modo de seleccionado.

a) Salida de voltaje

Para escribir un valor de voltaje en una salida analógica, colocar el número de canal que se va a utilizar seguido del valor que se desea escribir y de la palabra verdadero (true) para escribir dicho valor en la EEPROM del DAC caso contrario escribir falso (false). Para sacar voltaje por una salida analógica se tienen las siguientes opciones: Indio.analogWrite(1, 2.67,

true); //Fija CH1 a 2.67V ("true" escribe el valor en la EEPROM del DAC). Indio.analogWrite(1, 33.5, true); // Fija CH1 a 33.5% (aproximadamente en 3.685V). Indio.analogWrite(1, 1000, true); //Fija CH1 al valor 1000 del DAC (aproximadamente en 2.685V). (GitHub, Inc., 2016)

b) Salida de corriente

Para escribir un valor de corriente en una salida analógica, colocar el número de canal que se va a utilizar seguido del valor que se desea escribir y de la palabra verdadero (true) para escribir dicho valor en la EEPROM del DAC caso contrario escribir falso (false). Para sacar corriente por una salida analogica se tienen las siguientes opciones: Indio.analogWrite(1, 10.50, false); //Fija CH1 a 10.5 mA ("false" no escribe el valor en la EEPROM del DAC). Indio.analogWrite(1, 75, true); //Fija CH1 a 75% (aproximadamente 16 mA). Indio.analogWrite(1, 2048, true); //Fija CH1 a un valor de 2048 del DAC (aproximadamente 10.5 mA). (GitHub, Inc., 2016)

2.5. ¿Qué es un sensor?

De acuerdo con (Navas, S.f.) un sensor o captador es “un dispositivo diseñado para recibir información de una magnitud del exterior y transformarla en otra magnitud, normalmente eléctrica, capaz de cuantificarla y manipularla”.

2.5.1. Sensor de nivel

De acuerdo con (Omega Engineering, 2016) el sensor de nivel es “un dispositivo electrónico que mide la altura del material, dentro de un tanque u otro recipiente integral para el control de procesos en muchas industrias”.

2.5.2. Sensor de temperatura

De acuerdo con (Jeison Torres, 2010) un sensor de temperatura es “un dispositivo capaz de interpretar señales de cambio de temperaturas y transformar esta información en señales eléctricas y enviarlas a otro dispositivo para poder ser interpretadas”.

2.6. Medición de nivel

La medición de nivel se define como la determinación de la posición de la interface entre dos medios. Estos son usualmente fluidos, pero pueden existir sólidos o combinación de ellos. La interface puede existir entre un líquido y un gas, un líquido y su vapor, dos líquidos, un sólido o sólido diluido y un gas. (Medición de nivel de líquidos, 2011)

2.6.1. Clases de medición de nivel de líquidos

La medición de nivel se justifica tanto en mediciones de procesos continuos como en mediciones puntuales tales como alarmas por alto o bajo nivel. En procesos continuos se tiende al logro de una capacidad de almacenamiento menor, reduciendo el costo inicial del equipo y exigiendo la necesidad de un control preciso y sensible de nivel. La aplicación más frecuente de los detectores de nivel es la señalización de lleno y vacío. Para la medición continua se utilizan, entre otros, los principios: capacitivo, conductivo, vibratorio, a microondas, hidrostático, a ultrasonido y radiométrico, además, en la actualidad, para la investigación se utilizan los basados en elementos radioactivos. (Medición de nivel de líquidos, 2011)

2.6.2. Aplicaciones de las mediciones de nivel

Las técnicas utilizadas de medición de nivel están presentes en todos los campos entre los cuales se tienen: química, petroquímica, alimentación, cervecería, tratamiento de aguas blancas, tratamiento de aguas residuales, materiales para la construcción, rocas y minerales, centrales de energía, fabricación de papel, astilleros, industrias del automóvil y aeronáutica. Las razones para medir el nivel de líquidos son muy diversas, por lo general, una o más de las que se describen a continuación constituyen el motivo para la medición (y frecuentemente, el control automático) del nivel de líquidos: Mantenimiento de condiciones seguras de operación, la optimización y control de procesos, el ámbito de la investigación y el desarrollo entre otros. (Medición de nivel de líquidos, 2011)

2.6.3. Métodos de medición de nivel

De acuerdo con (Medición de nivel de líquidos, 2011) “Los métodos más utilizados en la industria para la medición de nivel de líquidos pueden ser clasificados en métodos de medición directa e indirecta”.

2.7. Método de medición directa de nivel

Los métodos de medición directa utilizan para la medición de nivel, la altura del líquido sobre una línea de referencia. Entre los métodos de medición directa se tienen los Indicadores visuales, los Indicadores de cristal y los instrumentos de flotador. (Medición de nivel de líquidos, 2011)

2.7.1. Indicador de cristal

El funcionamiento del indicador de cristal se basa en el principio de los vasos comunicantes: con igual presión, el líquido del tanque sube en el tubo de vidrio hasta que ambos niveles sean iguales. Cuando el nivel varía en el tanque, varía también en el tubo de vidrio obteniéndose así una indicación real de nivel del proceso. En la Figura 11 se representa este método de medición y se puede observar que sirve tanto para tanques abiertos como cerrados. (Medición de nivel de líquidos, 2011)

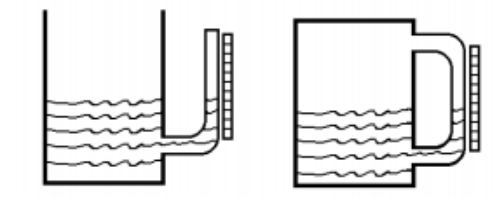


Figura 11 Medidores de cristal

Fuente: (Medición de nivel de líquidos, 2011)

2.7.2. Ventajas del indicador de cristal

De acuerdo con (Medición de nivel de líquidos, 2011) las ventajas del indicador de cristal son: “es económico, presenta seguridad en la lectura del nivel del líquido y es preciso”.

2.7.3. Desventajas del indicador de cristal

De acuerdo con (Medición de nivel de líquidos, 2011) las desventajas del indicador de cristal son: “se debe manipular con cuidado para evitar roturas, no es adecuado para control directo, su campo de medida es limitado, la indicación es local y es susceptible a ensuciarse por las características del líquido”.

2.7.4. Aplicaciones del indicador de cristal

De acuerdo con (Medición de nivel de líquidos, 2011) las aplicaciones del indicador de cristal son: “para medición de líquidos, lecturas periódicas de nivel en tanques abiertos y tanques cerrados”.

2.8. Método de medición indirecta de nivel

De acuerdo con (Medición de nivel de líquidos, 2011) entre los métodos de medición indirecta para la medición de nivel de líquidos se tienen “los que aprovechan el empuje producido por el propio líquido, la presión hidrostática y los que aprovechan las características del líquido”.

2.8.1. Medidor de nivel de ultrasonidos

> El sistema ultrasónico de medición de nivel se basa en la emisión de un impulso ultrasónico a una superficie reflectante y la recepción del eco del mismo en un receptor. El retardo en la captación del eco depende del nivel del tanque. (Creus, 2010)

2.8.1.1. Funcionamiento

En el medidor de nivel por ultrasonido, cuando las ondas sonoras viajan en un medio que absorbe el sonido y golpean a otro medio tal como una pared, una partícula en el líquido, o la superficie del líquido, solamente una pequeña porción de la energía de la onda sonora penetra la barrera y el resto de la energía se refleja. La onda sonora reflejada es un eco. El emisor dispone de un oscilador excitador para enviar un impulso ultrasónico a la superficie del fluido y el receptor, recibe esta señal reflejada enviando una señal función del tiempo transcurrido, y por lo tanto del nivel, a un indicador. En otras palabras, el nivel se mide en función del tiempo necesario para que la señal se desplace del transmisor a la superficie del líquido y retorne al receptor. Se utiliza la siguiente fórmula D=ct/2, donde: D es la distancia desde el sensor al objetivo, c es la velocidad del sonido en el aire, t es el tiempo de tránsito para el pulso ultrasónico. (Medición de nivel de líquidos, 2011)

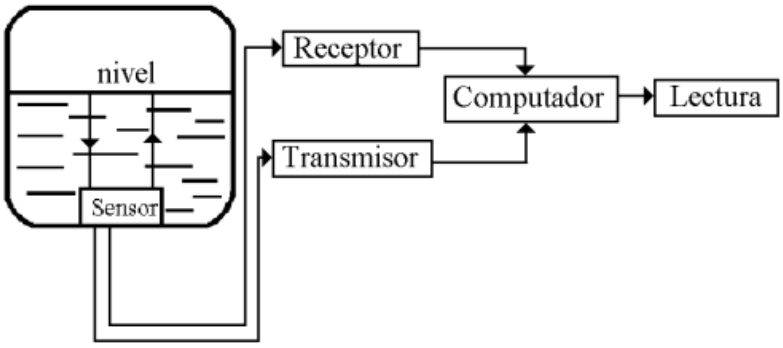


Figura 12 Diagrama de bloques del medidor de ultrasonido.

Fuente: (Medición de nivel de líquidos, 2011)

Para medición continua de nivel, los dispositivos deben ubicarse de manera tal, que la señal de ultrasonido apunte directamente al material para tener así la trayectoria de medición más directa. En la Figura 13 se muestran varias disposiciones de montaje de los sensores que se utilizan en casos de alarmas o de indicación continua de nivel. (Medición de nivel de líquidos, 2011)

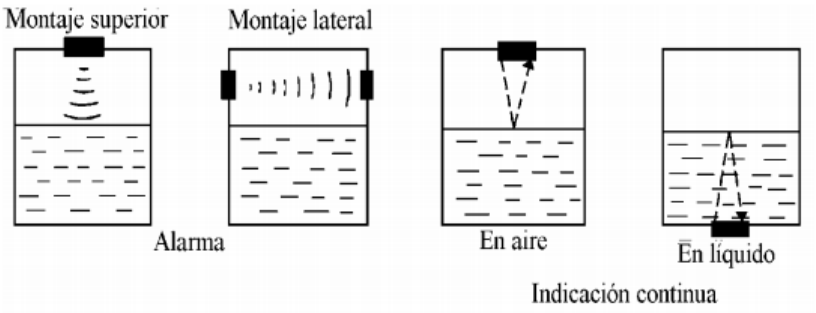


Figura 13 Montajes de los sensores para medidor de ultrasonido

Fuente: (Medición de nivel de líquidos, 2011)

2.8.1.2. Criterios para la selección del sistema de ultrasonidos

a) Distancia que se va a medir

Este es el primer factor que se debe considerar. Para elegir entre los varios tipos de amplificadores y sensores, debería consultarse una tabla de especificaciones y los datos del proceso. Un sistema desarrollado para obtener una precisión muy alta en distancias cortas no sería lo suficientemente eficaz para distancias largas. (Kamp, 2008)

b) Composición y propiedades de la superficie del producto

Es inherente a este tipo de sistemas que por lo menos una pequeña parte de la señal sonora transmitida se refleje en la superficie del producto. Si en la superficie del líquido hay algún tipo de capa o espuma o si el producto es algún árido compuesto de pequeños gránulos, se reflejara menos energía y se requerirá un mayor poder de transmisión para la misma distancia. (Kamp, 2008)

c) Condiciones para la medición

Tanto la señal emitida desde el sensor como la reflejada deben atravesar la atmosfera del tanque o silo. Para elegir el sistema de medición más adecuado, se deben examinar cuidadosamente todos los factores susceptibles de poder amortiguar la señal o que puedan provocar interferencias por absorción o reflexión. Estos factores incluyen la presencia de polvo, vapor, presión, temperatura y cambios en la composición de los gases. (Kamp, 2008)

d) Montaje

El sensor se debe montar en la mejor posición posible para que el sistema funcione correctamente, donde no haya obstáculos entre la superficie del producto y el sensor. Las escaleras de acceso, los elementos de inmersión, los removedores, las entradas de bombeo o una entrada del producto que caigan pueden provocar interferencias. (Kamp, 2008)

2.8.1.3. Aplicaciones

Los sistemas de medición por ultrasonidos se pueden emplear para una gran variedad de aplicaciones, desde cambios en el nivel del agua de menos de 10 o 15 cm para mediciones de caudal hasta la determinación de la cantidad de carbón o caliza en un silo de 60 m de profundidad. Es evidente que con tal variedad de aplicaciones, se requieren distintos tipos de sensores y de los correspondientes amplificadores. El aspecto más relevante para la medición cuando se requiere detectar pequeñas variaciones de nivel en la superficie de un líquido en calma es la precisión, mientras que si se pretende medir distancias largas en condiciones desfavorables, lo primero que hay que exigir es un alto grado de fiabilidad en la medición. (Kamp, 2008)

2.9. Medición de temperatura

La medida de temperatura constituye una de las medidas más comunes y más importantes que se efectúan en los procesos industriales. Las limitaciones del sistema de medida quedan definidas en cada tipo de aplicación por la precisión, por la velocidad de captación de la temperatura, por la distancia entre el elemento de medida y el aparato receptor y por el tipo de instrumento indicador, registrador o controlador. Los instrumentos de temperatura utilizan diversos fenómenos que son influidos por la temperatura y entre los cuales figuran: la variación de resistencia de un conductor (sonda de resistencia), la variación de resistencia de un semiconductor (termistores), la fuerza electromotriz (f.e.m.) creada en la unión de dos metales distintos (termopares o termocuplas) y la intensidad de la radiación total emitida por el cuerpo (pirómetros de radiación). (Beltrán, 2011)

2.9.1. RTDs

La medida de temperatura utilizando sondas de resistencia depende de las características de resistencia en función de la temperatura que son propias del elemento de detección. El elemento consiste usualmente en un arrollamiento de hilo muy fino del conductor adecuado bobinado entre capas de material aislante y protegido con un revestimiento de vidrio o de cerámica. El material que forma el conductor se caracteriza por el llamado “coeficiente de temperatura de resistencia” que expresa a una temperatura especificada, la variación de la resistencia en ohmios del conductor por cada grado que cambia su temperatura. (Beltrán, 2011)

2.9.2. Termistores

Los termistores son semiconductores electrónicos con un coeficiente de temperatura de resistencia negativo de valor elevado, por lo que presentan unas variaciones rápidas, y extremadamente grandes, para los cambios, relativamente pequeños, en la temperatura. Los termistores se fabrican con óxidos de níquel, manganeso, hierro, cobalto, cobre, magnesio, titanio y otros metales, y están encapsulados en sondas y en discos. Los

termistores también se denominan NTC (Negative Temperature Coeficient - coeficiente de temperatura negativo) existiendo casos especiales de coeficiente positivo cuando su resistencia aumenta con la temperatura (PTC - Positive Temperature Coeficient). (Creus, 2010)

2.9.3. Termopares o Termocuplas

Un termopar es un dispositivo capaz de convertir la energía calorífica en energía eléctrica. Su funcionamiento se basa en los descubrimientos hechos por Thomas Seebeck en 1821 cuando hizo circular corriente eléctrica en un circuito, formado por dos metales diferentes cuyas uniones se mantienen a diferentes temperaturas, esta circulación de corriente obedece a dos efectos termoeléctricos combinados, el efecto Peltier que provoca la liberación o absorción de calor en la unión de dos metales diferentes cuando una corriente circula a través de la unión y el efecto Thompson que consiste en la liberación o absorción de calor cuando una corriente circula a través de un metal homogéneo en el que existe un gradiente de temperaturas. (Beltrán, 2011)

2.9.3.1. Efecto Seebeck

El efecto Seebeck es una propiedad termoeléctrica descubierta en 1821 por el físico alemán Thomas Johann Seebeck inversa al efecto Peltier. Este efecto provoca la conversión de una diferencia de temperatura en electricidad. Se crea un voltaje en presencia de una diferencia de temperatura entre dos metales o semiconductores homogéneos. Una diferencia de temperaturas T1 y T2 en las juntas entre los metales A y B induce una diferencia de potencial V. (Tecnomedición, 2010)

2.9.4. Termopozos

Se utilizan para proteger los sensores de temperatura, tales como termopares, termistores y termómetros bimetálicos, contra los daños causados por presión excesiva, velocidad del material y corrosión. También aumentan la longevidad del sensor, permiten la sustitución del sensor sin necesidad de vaciar el sistema y reducen la probabilidad de contaminación. Están diseñados para aplicaciones de alta presión normalmente se mecanizan a partir de barras para garantizar su integridad. (Omega Engineering, 2017)

2.9.4.1. Tipos de termopozos

Los termopozos se clasifican de acuerdo con el diseño del vástago. Un termopozo recto tiene el mismo diámetro en toda su longitud de inserción y ofrece protección contra la corrosión y la erosión. Termopozos escalonados generalmente tienen un diámetro de ¾ " en la parte superior, que se reduce a ½" de diámetro cerca de la punta. El área reducida de superficie permite velocidades más suaves y respuestas más rápidas a la temperatura para dispositivos de detección. Termopozos cónicos tienen un diámetro que disminuye gradualmente a lo largo de la longitud de inserción. Ellos ofrecen una resistencia superior, así como tiempos de

respuesta rápidos a los cambios de temperatura. Termopozos cónicos se utilizan con mayor frecuencia en aplicaciones de alta velocidad. Los estudios de caso realizados con termopozos rectos y cónicos utilizados en tuberías de gas natural revelaron que los termopares rectos sufrían una falla prematura cuando eran expuestos a vibraciones inducidas de flujo. (Omega Engineering, 2017)

Figura 14 Tipos de termopozos

Fuente: (Omega Engineering, 2017)

2.9.4.2. Tipo de conexión

Los termopozos pueden ser conectados a un RTD, termistor o termopar por medio de varios tipos diferentes de conexión. Algunos de los más comunes son: roscado (Threaded), soldable (Weld-in), para conexión soldable (Socket weld), junta tórica (O-ring), bridado (Flange) y de abrazadera (Clamp). Las conexiones roscadas están hechas de materiales que pueden ser soldados y proporcionan una resistencia adicional. En procesos en los que los contaminantes de los hilos se deben evitar, tales como en las industrias alimentarias y farmacéuticas, se utilizan comúnmente las conexiones soldables. Las conexiones de O-ring utilizan una junta tórica para sellar el interior de un manguito soldado a un tanque. (Omega Engineering, 2017)

2.10. Transmisores

Los transmisores son instrumentos que transmiten a distancia la variable del proceso en forma de señal neumática, electrónica, digital, etc. Así la señal neumática es de 3 a 15 psi (libras por pulgada cuadrada), la señal electrónica normalizada de corriente es de 4 a 20 mA o de 0 a 20 mA y la de voltaje es de 1 a 5 V o de 0 a 10 V de corriente continua y la señal digital consiste en una serie de impulsos en forma de bits. (Creus, 2010)

2.10.1. Transmisores electrónicos

Basados en detectores de inductancia, o utilizando transformadores diferenciales o circuitos de puente de Wheatstone, o empleando una barra de equilibrio de fuerzas, convierten la señal de la variable a una señal

electrónica de 4-20 mA c.c. Su exactitud es del orden del ± 0,5%. (Creus, 2010)

2.11. Comunicación serial

La comunicación serial consiste en el envío de un bit de información de manera secuencial, esto es, un bit a la vez y a un ritmo acordado entre el emisor y el receptor. La comunicación serial en computadores ha seguido los estándares definidos en 1969 por el RS-232 (Recommended Standard 232) que establece niveles de voltaje, velocidad de transmisión de los datos, etc. Por ejemplo, este protocolo establece un nivel de -12v como un uno lógico y un nivel de voltaje de +12v como un cero lógico (por su parte, los microcontroladores emplean por lo general 5v como un uno lógico y 0v como un cero lógico). Existen en la actualidad diferentes ejemplos de puertos que comunican información de manera serial (un bit a la vez). El conocido como "puerto serial" ha sido gradualmente reemplazado por el puerto USB (Universal Serial Bus) que permite mayor versatilidad en la conexión de múltiples dispositivos. (Tamayo, 2009)

2.11.1. Comunicación serial en Arduino

La mayoría de los micro controladores, entre ellos Arduino, poseen un puerto de comunicación serial. Para comunicarse con los computadores personales actuales que poseen únicamente puerto USB requieren de un dispositivo "traductor". Arduino emplea el integrado FT232R, el cual es un convertidor USB-Serial. A través de este integrado el microcontrolador puede recibir y enviar datos a un computador de manera serial. La parte física encargada de la comunicación serial es la UART (Universal Asynchronous Receiver and Transmitter). Los micro controladores Atmega8/168/328, en los cuales está basado Arduino, disponen de un dispositivo compatible llamado USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) que permite tanto la comunicación asincrónica como sincrónica. (Tamayo, 2009)

2.11.2. Comunicación asincrónica/sincrónica

En la comunicación asincrónica, la velocidad de envío de los datos es acordada a prioridad entre el emisor y el receptor. En la comunicación sincrónica, el envío de los datos es sincronizado por el emisor a partir de un pulso constante de reloj (Clock), con cada pulso envía un nuevo dato. (Tamayo, 2009)

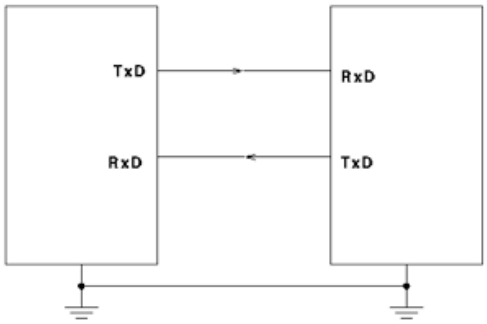


Figura 15 Comunicación asincrónica

Fuente: (Tamayo, 2009)

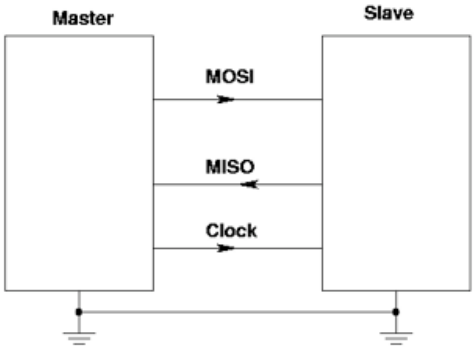


Figura 16 Comunicación sincrónica

Fuente: (Tamayo, 2009)

2.11.3. Comunicación serial entre el computador y Arduino

En la comunicación con el computador, Arduino emplea la comunicación asincrónica. Para esto, requiere dos líneas de conexión y establecer un nivel de tierra común con el computador, estableciendo el mismo nivel de voltaje de referencia. Además de realizar las conexiones físicas entre el microcontrolador y el computador, para que pueda establecerse la comunicación serial debe existir un acuerdo previo en la manera cómo van a ser enviados los datos. Este acuerdo debe incluir los niveles de voltaje que serán usados, el tamaño y formato de cada uno de los mensajes (número de bits que constituirán el tamaño de la palabra, existirá o no un bit de inicio y/o de parada, se empleará o no un bit de paridad), el tipo de lógica empleada (qué voltaje representará un cero o un uno), el orden en que serán enviados los datos (será enviado primero el bit de mayor peso o el de menor peso) y la velocidad de envío de datos. (Tamayo, 2009)

2.11.3.1. Velocidad de transmisión

Arduino facilita este proceso para que sólo sea necesario especificar la velocidad de envío de los datos. Esta velocidad es conocida como "baud rate" o rata de pulsos por segundo. Velocidades frecuentes de uso son 9600, 19200, 57600 y 115200. (Tamayo, 2009)

2.11.3.2 Interfaz gráfica de Arduino

La interfaz gráfica de Arduino presenta un "monitor serial" donde pueden observarse los datos recibidos por el computador a través del puerto USB. En este monitor debe especificarse la velocidad a la cual el microcontrolador está enviando los datos, de tal manera que el computador pueda leer el puerto a esa misma velocidad. (Tamayo, 2009)

2.12. HMI

HMI (Human Machine Interface) o interfaz hombre máquina, es el dispositivo o sistema que permite el interfaz entre la persona (operador) y la máquina (proceso). Tradicionalmente estos sistemas consistían en paneles compuestos por indicadores y comandos, tales como luces pilotos, indicadores digitales y análogos, etc. En la actualidad, es posible contar con sistemas de HMI bastantes más poderosos y eficaces, además de permitir una conexión más sencilla y económica con el proceso o máquinas. (Turmero, S.f.)

2.12.1. Tipos

2.12.1.1. Terminal de Operador

De acuerdo con (Turmero, S.f.) el terminal de operador consistente en "un dispositivo, generalmente construido para ser instalado en ambientes agresivos, donde pueden ser solamente de despliegues numéricos, o alfanuméricos o gráficos. Pueden ser además con pantalla sensible al tacto (touch screen)".

2.12.1.2. PC + Software

Constituye otra alternativa basada en un PC en donde se carga un software apropiado para la aplicación. Como PC se puede utilizar cualquiera según lo exija el proyecto, en donde existen los llamados Industriales (para ambientes agresivos), los de panel (Panel PC) que se instalan en gabinetes dando una apariencia de terminal de operador, y en general muchas formas de hacer un PC, pasando por el tradicional PC de escritorio. (Turmero, S.f.)

2.12.2. Software HMI

El software permite entre otras cosas las siguientes funciones: interfaz gráfica con la finalidad de poder ver el proceso, registro en tiempo real e histórico de datos y manejo de alarmas. Al igual que en los terminales de operador, se requiere de una herramienta de diseño o desarrollo, la cual se usa para configurar la aplicación deseada, y luego debe quedar corriendo en el controlador un software de ejecución (Run Time). Por otro lado, este software puede comunicarse directamente con los dispositivos externos (proceso) o bien hacerlo a través de un software especializado en la comunicación. (Turmero, S.f.)

2.12.3 Desarrolladores de software HMI

De acuerdo con (Turmero, S.f.) los desarrolladores de software HMI son: "National Instruments (LabVIEW), Siemens (WinCC), GE Fanuc (IFIX / Cimplicity), Omrom (SCS), Allen-Bradley (RS-View) y Wonderware (InTouch)".

2.13. LabVIEW

LabVIEW es un entorno de desarrollo integrado y diseñado específicamente para ingenieros y científicos. Nativo de LabVIEW es un lenguaje de programación gráfica (G) que utiliza un modelo de flujo de datos en lugar de líneas secuenciales de código de texto, lo que le permite escribir código funcional utilizando un diseño visual que se asemeja a su proceso de pensamiento. Esto significa que usted emplea menos tiempo preocupándose por el por punto y coma y la sintaxis y más tiempo resolviendo los problemas que importan. (National Instruments, 2016)

Figura 17 Software LabVIEW 2014

Fuente: (National, 2014)

2.13.1. Funcionamiento y estructura

Los programas desarrollados mediante LabVIEW se denominan Instrumentos Virtuales (VIs), porque su apariencia y funcionamiento imitan los de un instrumento real. Sin embargo son análogos a las funciones creadas con los lenguajes de programación convencionales. Los VIs tienen una parte interactiva con el usuario y otra parte de código fuente, y aceptan parámetros procedentes de otros VIs. Todos los VIs tienen un panel frontal y un diagrama de bloques. Las paletas contienen las opciones que se emplean para crear y modificar los VIs. (Tutorial LabVIEW, 2009)

2.13.1.1. Panel Frontal

Se trata de la interfaz gráfica del VI con el usuario. Esta interfaz recoge las entradas procedentes del usuario y representa las salidas proporcionadas por el programa. Un panel frontal está formado por una serie de botones, pulsadores, potenciómetros, gráficos, etc. Cada uno de ellos puede estar definido como un control o un indicador. Los primeros sirven para introducir parámetros al VI, mientras que los indicadores se emplean para mostrar los resultados producidos, ya sean datos adquiridos o resultados de alguna operación. (Tutorial LabVIEW, 2009)

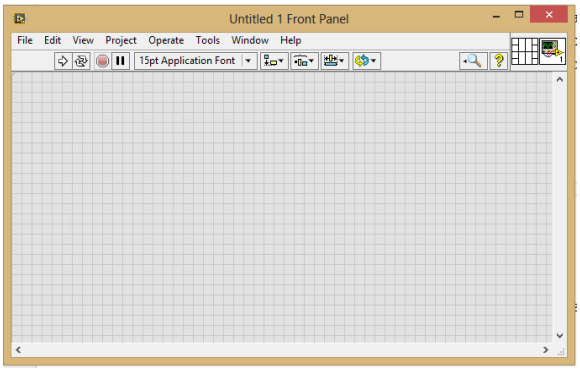


Figura 18 Panel frontal de LabVIEW

Fuente: (National, 2014)

2.13.1.2. Diagrama de bloques

El diagrama de bloques constituye el código fuente del VI. En el diagrama de bloques es donde se realiza la implementación del programa del VI para controlar o realizar cualquier procesado de las entradas y salidas que se crearon en el panel frontal. El diagrama de bloques incluye funciones y estructuras integradas en las librerías que incorpora LabVIEW. Los controles e indicadores que se colocaron previamente en el Panel Frontal, se materializan en el diagrama de bloques mediante los terminales. El diagrama de bloques se construye conectando los distintos objetos entre sí, como si de un circuito se tratara. Los cables unen terminales de entrada y salida con los objetos correspondientes, y por ellos fluyen los datos. LabVIEW posee una extensa biblioteca de funciones, entre ellas,

aritméticas, comparaciones, conversiones, funciones de entrada/salida, de análisis, etc. Las estructuras, similares a las declaraciones causales y a los bucles en lenguajes convencionales, ejecutan el código que contienen de forma condicional o repetitiva (bucle for, while, case,...). Los cables son las trayectorias que siguen los datos desde su origen hasta su destino, ya sea una función, una estructura, un terminal, etc. (Tutorial LabVIEW, 2009)

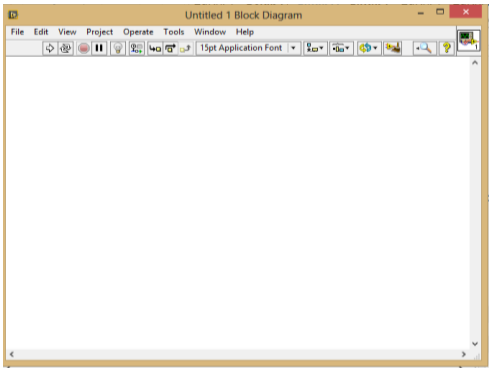


Figura 19 Diagrama de bloques

Fuente: (National, 2014)

2.13.1.3 Paletas

El entorno de LabVIEW dispone de paletas que proporcionan herramientas necesarias para desarrollar instrumentos virtuales, estas paletas están disponibles tanto en el panel frontal como en el diagrama de bloques.

a) Paleta de herramientas

De acuerdo con (Tutorial LabVIEW, 2009) la paleta de herramientas se emplea en "el panel frontal como en el diagrama de bloques. Contiene las herramientas necesarias para editar y depurar los objetos tanto del panel frontal como del diagrama de bloques."

Figura 20 Paleta de herramientas

Fuente: (National, 2014)

b) Paleta de controles

De acuerdo con (Tutorial LabVIEW, 2009) la paleta de controles se utiliza únicamente en “el panel frontal y contiene todos los controles e indicadores que se emplearán para crear la interfaz del VI con el usuario”.

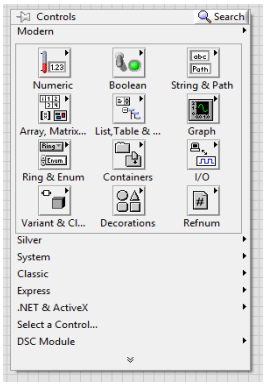


Figura 21 Paleta de controles

Fuente: (National, 2014)

c) Paleta de funciones

Se emplea en el diseño del diagrama de bloques. La paleta de funciones contiene todos los objetos que se emplean en la implementación del programa del VI, ya sean funciones aritméticas, de entrada/salida de señales, entrada/salida de datos a fichero, adquisición de señales, temporización de la ejecución del programa. (Tutorial LabVIEW, 2009)

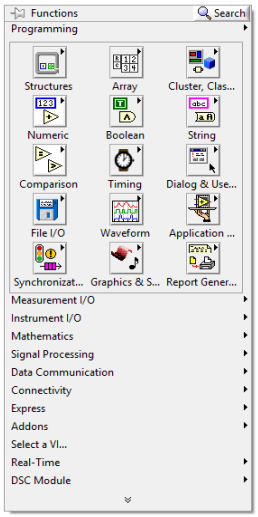


Figura 22 Paleta de funciones

Fuente: (National, 2014)

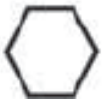
2.14. P&ID

Diagrama de instrumentación y tubería, por sus siglas en ingles P&ID, (Piping and Instrument Diagram). Son documentos, base de todo proyecto industrial, que muestran el flujo de proceso en las tuberías, así como los equipos instalados y el instrumental. En estos diagramas se muestra como mínimo la siguiente información: equipos de planta, las tuberías que interconectan los equipos y la instrumentación que controla la planta. Los diagramas a medida que van pasando por las distintas etapas de un proyecto, se van completando con información cada vez más detallada, que permiten finalmente reflejar el funcionamiento y las características principales de la planta. (Uriza, 2016)

2.14.1. Identificación y símbolos de instrumentación

2.14.1.1. Designación de instrumentos

De acuerdo con (Uriza, 2016) la designación es “para representar un instrumento en un plano de instrumentación para esto se debe de utilizar un círculo. Para el caso donde el círculo está dentro de un cuadrado, simboliza un instrumento que comparte un display o un control”.

	Ubicación primaria. Accesible normalmente al operador	Montado en campo	Ubicación Auxiliar. Accesible normalmente al operador
Instrumentos discretos			
Visualización compartida, Control compartido			
Función de ordenador			

Continua

Controlador lógico programable (PLC)			

Figura 23 Símbolos generales de instrumentos

Fuente: (Creus, 2010)

2.14.1.2. Identificación de instrumentos

La identificación de instrumentos cuando se emplean solo dos letras, la primera letra siempre designa a la variable a la que está dedicada el instrumento. La segunda es la función principal del instrumento. Cuando se emplean tres letras, la primera letra designa la variable, la segunda letra es la función secundaria y la tercera letra es la función principal. Ejemplos: TT (Transmisor de temperatura), LIC (Controlador e indicador de nivel). (Páez, 2004)

Tabla 1

Código de identificación de instrumentos

PRIMERA LETRA			**LETRAS SUCESIVAS**		
	Variable medida o inicial	Letra de modificación	Lectura o función de lectura pasiva	Función de salida	Letra de modificación
A	Análisis		Alarma		
B	Quemador, combustión		Libre	Libre	Libre
C	Libre			Control	
D	Libre	Diferencial			
E	Tensión		Sensor		
F	Caudal	Relación			

Continua

G	Libre		Vidrio		G
H	Manual				Alto
I	Corriente		Indicar		
J	Potencia	Exploración			
K	Tiempo	Variación de tiempo		Estación de control	
L	Nivel		Luz		Bajo
M	Libre	Momentáneo			Medio, intermedio
N	Libre		Libre	Libre	Libre
O	Libre		Orificio, restricción		
P	Presión, vacío		Punto (ensayo) conexión		
Q	Cantidad	Integrar totalizar			
R	Radiación		Registro		
S	Velocidad, frecuencia	Seguridad		Interruptor	
T	Temperatura			Transmisión	
U	multivariable		multifunción	Multifunción	Mltifunción
V	Vibración, análisis mecánico			Válvula, regulador tiro, persiana	
W	Peso, fuerza		Vaina, sonda		
X	Sin clasificar	Eje x	Sin clasificar	Sin clasificar	Sin clasificar

Continua

Y	Evento, estado o presencia	Eje y	Relé, calculo, conversión
Z	Posición, dimensión	Eje z	Motor, actuador, elemento final de control sin clasificar

Fuente: (Creus, 2010)

2.14.1.3. Simbología

a) Representación de líneas

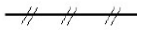
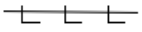
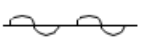
Símbolo	Significado
	Tubería de proceso (línea de proceso)
	Señal neumática
	Señal eléctrica
	Señal hidráulica
	Señal acústica o electromagnética

Figura 24 Representación de líneas

Fuente: (Páez, 2004)

b) Cuerpos de válvulas

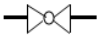
Símbolo	Significado
	Cuerpo general
	Válvula mariposa
	Válvula de bola
	Válvula de globo
	Válvula normal abierta (común)
	Válvula normal cerrada (común)

Figura 25 Representación de cuerpos de válvulas

Fuente: (Páez, 2004)

c) Actuadores

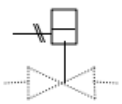
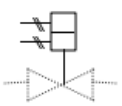
Símbolo	Significado
	Diafragma neumático
	Motorizado
	Solenoide
	Cilindro contra resorte
	Cilindro doble acción

Figura 26 Representación de actuadores

Fuente: (Páez, 2004)

2.15. Sistema de control

Un sistema de control es un tipo de sistema que se caracteriza por la presencia de una serie de elementos que permiten influir en el funcionamiento del sistema. La finalidad de un sistema de control es conseguir, mediante la manipulación de las variables de control, un dominio sobre las variables de salida, de modo que estas alcancen unos valores prefijados (consigna). (Sistemas de control, 2016)

2.15.1. Sistema de control de lazo abierto

Son los sistemas en los cuales la salida no afecta la acción de control. En un sistema en lazo abierto no se mide la salida ni se realimenta para compararla con la entrada. En cualquier sistema de control en lazo abierto, la salida no se compara con la entrada de referencia. Por tanto, a cada entrada de referencia le corresponde una condición operativa fija; como resultado, la precisión del sistema depende de la calibración. Ante la presencia de perturbaciones, un sistema de control en lazo abierto no realiza la tarea deseada. En la práctica, el control en lazo abierto sólo se utiliza si se conoce la relación entre la entrada y la salida y si no hay perturbaciones internas ni externas. (Salvetti, 2012)

2.15.1.1. Elementos básicos

Los elementos básicos de un sistema de control de lazo abierto son: el elemento de control: este elemento determina qué acción se va a tomar dada una entrada al sistema de control. El elemento de corrección: este elemento responde a la entrada que viene del elemento de control e inicia la acción para producir el cambio en la variable controlada al valor requerido. El proceso: o planta es el sistema en el que se va a controlar la variable. (Salvetti, 2012)

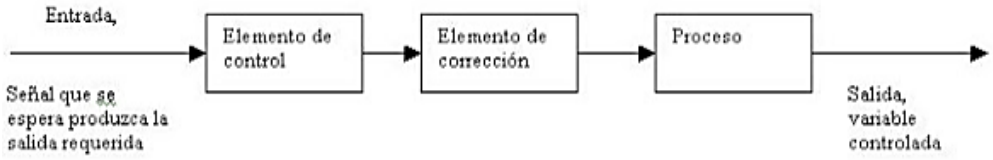


Figura 27 Sistema de control de lazo abierto

Fuente: (Salvetti, 2012)

2.15.2. Sistema de control de lazo cerrado

Los sistemas de control realimentados se denominan también sistemas de control de lazo cerrado. En un sistema de control en lazo cerrado, se alimenta al controlador la señal de error de actuación, que es la diferencia entre la señal de entrada y la salida de realimentación (que puede ser la señal de salida misma o una función de la señal de salida y sus derivadas

o integrales) a fin de reducir el error y llevar la salida del sistema a un valor conveniente. El término control en lazo cerrado siempre implica el uso de una acción de control realimentando para reducir el error del sistema. (Salvetti, 2012)

2.15.2.1. Elementos básicos

Los elementos básicos de un sistema de control de lazo abierto son: el elemento de comparación: este elemento compara el valor requerido o de referencia de la variable por controlar con el valor medido de lo que se obtiene a la salida, y produce una señal de error la cual indica la diferencia del valor obtenido a la salida y el valor requerido. El elemento de control: este elemento decide que acción tomar cuando se recibe una señal de error. El elemento de corrección: este elemento se utiliza para producir un cambio en el proceso al eliminar el error. El elemento de proceso: el proceso o planta, es el sistema dónde se va a controlar la variable. El elemento de medición: este elemento produce una señal relacionada con la condición de la variable controlada, y proporciona la señal de realimentación al elemento de comparación para determinar si hay o no error. (Salvetti, 2012)

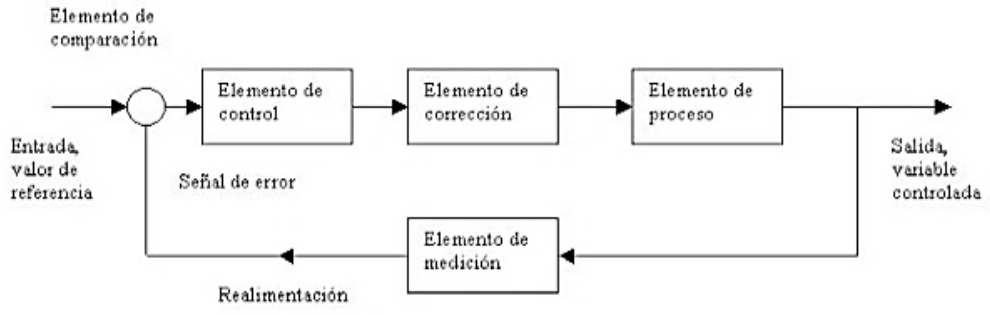


Figura 28 Sistema de control de lazo cerrado

Fuente: (Salvetti, 2012)

2.16. Elementos finales de control

Los elementos finales de control son los dispositivos encargados de transformar una señal de control en un flujo de masa o energía (variable manipulada). Es esta variable manipulada la que incide en el proceso causando cambios de la variable controlada. (Control de procesos–Facet–Unt, S.f.)

2.16.1. Válvulas

Las válvulas son unos de los instrumentos de control más esenciales en la industria. Debido a su diseño y materiales, las válvulas pueden abrir y cerrar, conectar y desconectar, regular, modular o aislar una enorme serie de líquidos y gases, desde los más simples hasta los más corrosivos o tóxicos. (Caroli, S.f.)

2.16.1.1. Válvula de solenoide

La válvula de solenoide es un dispositivo operado eléctricamente, y es utilizado para controlar el flujo de líquidos o gases en posición completamente abierta o completamente cerrada. A diferencia de las válvulas motorizadas, las cuales son diseñadas para operar en posición moduladora, la válvula de solenoide no regula el flujo aunque puede estar siempre completamente abierta o completamente cerrada. La válvula de solenoide es una válvula que se cierra por gravedad, por presión o por la acción de un resorte; y es abierta por el movimiento de un émbolo operado por la acción magnética de una bobina energizada eléctricamente, o viceversa. Una válvula de solenoide consiste de dos partes accionantes distintas, pero integrales: un solenoide (bobina eléctrica) y el cuerpo de la válvula. (Válvulas de solenoide, 2016)

2.16.1.2. Principio de operación

El la figura 29 puede apreciarse las partes principales ya integradas de una válvula de solenoide típica. La aguja de la válvula está unida mecánicamente a la parte inferior del émbolo. En esta válvula en particular, cuando se energiza la bobina, el émbolo es levantado hacia el centro de la bobina, levantando la aguja del orificio donde está sentada, permitiendo así el flujo. Cuando se des energiza la bobina, el peso del émbolo hace que caiga por gravedad y cierre el orificio, deteniendo el flujo. En algunos tipos de válvulas, un resorte empuja el émbolo para que cierre la válvula; esto permite que la válvula pueda instalarse en otras posiciones diferentes a la vertical. (Válvulas de solenoide, 2016)

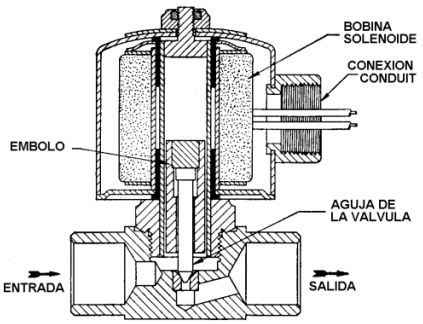


Figura 29 Partes de la válvula de solenoide

Fuente: (Válvulas de solenoide, 2016)

2.16.2. Bombas

Las bombas son dispositivos que se encargan de transferir energía a la corriente del fluido impulsándolo, desde un estado de baja presión estática a otro de mayor presión. Están compuestas por un elemento rotatorio denominado impulsor, el cual se encuentra dentro de una carcasa llamada voluta. Inicialmente la energía es transmitida como energía mecánica a

través de un eje, para posteriormente convertirse en energía hidráulica. El fluido entra axialmente a través del ojo del impulsor, pasando por los canales de éste y suministrándosele energía cinética mediante los álabes que se encuentran en el impulsor para posteriormente descargar el fluido en la voluta, el cual se expande gradualmente, disminuyendo la energía cinética adquirida para convertirse en presión estática. (Daniel Mogollón, 2010)

2.16.2.1. Bomba centrifuga

Una bomba centrífuga es una máquina que consiste de un conjunto de paletas rotatorias encerradas dentro de una caja o cárter, o una cubierta o coraza. Se denominan así porque la cota de presión que crean es ampliamente atribuible a la acción centrífuga. Las paletas imparten energía al fluido por la fuerza de esta misma acción. Así, una bomba centrífuga tiene dos partes principales:
Un elemento giratorio, incluyendo un impulsor y una flecha.
Un elemento estacionario, compuesto por una cubierta. (Daniel Mogollón, 2010)

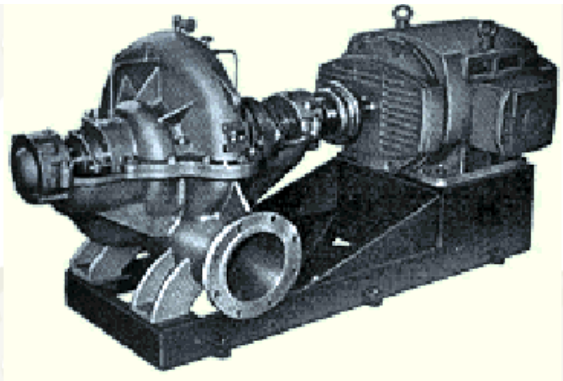


Figura 30 Bomba centrífuga

Fuente: (Daniel Mogollón, 2010)

2.16.2.2. Funcionamiento

El flujo entra a la bomba a través del centro u ojo del rodete y el fluido gana energía a medida que las paletas del rodete lo transportan hacia fuera en dirección radial. Esta aceleración produce un apreciable aumento de energía cinética y de presión, lo cual es debido a la forma de caracol de la voluta para generar un incremento gradual en el área de flujo. (Daniel Mogollón, 2010)

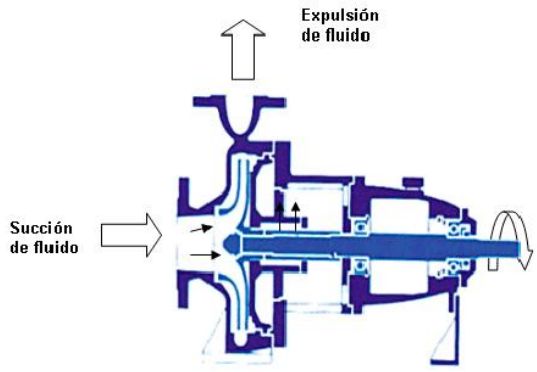


Figura 31 Funcionamiento de una bomba centrífuga

Fuente: (Daniel Mogollón, 2010)

2.16.2.3. Cavitación

La cavitación es un fenómeno físico, mediante el cual un líquido, en determinadas condiciones, pasa a estado gaseoso y unos instantes después pasa nuevamente al estado líquido. Este fenómeno tiene dos fases: cambio de estado líquido ha estado gaseoso y cambio de estado gaseoso a estado líquido. La cavitación es un fenómeno muy frecuente en sistemas hidráulicos donde se dan cambios bruscos de la velocidad del líquido. Los efectos que produce la cavitación son: ruidos y golpeteos, vibraciones y erosiones del material (daños debido a la cavitación). (Uralita Sistemas de tubería, S.f.)

2.16.2.4. Golpe de ariete

Se denomina golpe de ariete al choque violento que se produce sobre las paredes de un conducto forzado, cuando el movimiento líquido es modificado bruscamente. En otras palabras, el golpe de ariete se puede presentar en una tubería que conduzca un líquido hasta el tope, cuando se tiene un frenado o una aceleración en el flujo; por ejemplo, el cambio de abertura en una válvula en la línea. Al cerrarse rápidamente una válvula en la tubería durante el escurrimiento, el flujo a través de la válvula se reduce, lo cual incrementa la carga del lado aguas arriba de la válvula, iniciándose un pulso de alta presión que se propaga en la dirección contraria a la del escurrimiento. Esta onda provoca sobrepresiones y depresiones las cuales deforman las tuberías y eventualmente la destruyen. Desde el punto de vista energético puede considerarse la transformación de la energía cinética del fluido en energía potencial elástica (cambios de presión) y viceversa. Si la tubería carece de roce y es indeformable y por lo tanto no hay pérdidas de energía, el fenómeno se reproduce indefinidamente. Si hay roce y la tubería es elástica parte de la energía se va perdiendo y las sobrepresiones son cada vez menores hasta que el fenómeno se extingue. (Orozco, 2001)

2.16.3. Resistencia calefactora

Las resistencias de inmersión están diseñadas para el calentamiento en contacto directo con el fluido: agua, aceite, materiales viscosos, disoluciones ácidas o básicas. Dado que todo el calor se genera dentro del líquido, se alcanza un rendimiento energético máximo. Al no existir elementos distorsionadores, el control de la temperatura de proceso puede ser muy ajustado. Las resistencias de inmersión presentan varias opciones de acoplamiento al depósito o tanque donde se instalan: mediante tapón roscado, con racores, con brida, tipo sumergidores, etc. (Electricfor, S.f.)

2.17. Relé

El relé o relevador es un dispositivo electromecánico. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Dado que el relé es capaz de controlar un circuito de salida de mayor potencia que el de entrada, puede considerarse, en un amplio sentido, como un amplificador eléctrico. Como tal se emplearon en telegrafía, haciendo la función de repetidores que generaban una nueva señal con corriente procedente de pilas locales a partir de la señal débil recibida por la línea. Se les llamaba “relevadores”. De ahí el término “relé”. (Blog Tecnosinergia, 2014)

2.17.1 Estructura y funcionamiento

El electroimán hace bascular la armadura al ser excitada, cerrando los contactos dependiendo de si es NO o NC (normalmente abierto o normalmente cerrado). Si se le aplica un voltaje a la bobina se genera un campo magnético, que provoca que los contactos hagan una conexión. Estos contactos pueden ser considerados como el interruptor, que permite que la corriente fluya entre los dos puntos que cerraron el circuito o abriendo un circuito. La gran ventaja de los relés electromagnéticos es la completa separación eléctrica entre la corriente de accionamiento, la que circula por la bobina del electroimán, y los circuitos controlados por los contactos, lo que hace que se puedan manejar altos voltajes o elevadas potencias con pequeñas tensiones de control. También ofrecen la posibilidad de control de un dispositivo a distancia mediante el uso de pequeñas señales de control. Con estos modernos sistemas los relés pueden actuar de forma programada e independiente lo que supone grandes ventajas en su aplicación aumentando su uso en aplicaciones sin necesidad de utilizar controles como PLD’s u otros medios para comandarlos. Se puede encender por ejemplo una bombilla o motor y al encenderlo se apaga el otro motor o bombilla. (Blog Tecnosinergia, 2014)

CAPITULO III

DESARROLLO DEL TEMA

3.1. Características del proceso

- Fluido: agua.
- El nivel mínimo: 10 cm y el nivel máximo: 20 cm.
- Temperatura máxima del agua en el tanque de proceso: 40°C.
- El llenado del tanque no produce turbulencia ni espuma.
- El fluido del proceso no es corrosivo ni tampoco abrasivo.

3.2. Operación del proceso.

El fluido es suministrado desde un reservorio y en su trayecto al tanque pasa a través de un intercambiador de calor el cual baja la temperatura del fluido, en el tanque una vez que el fluido alcanza cierto nivel máximo se empieza a calentar el líquido hasta alcanzar una temperatura máxima, una vez alcanzada la temperatura deseada se comienza a drenar el fluido que cae por gravedad hacia el reservorio, cuando el nivel del fluido del tanque baja al nivel mínimo se deja de drenar el líquido y se comienza nuevamente a enviarlo hacia el tanque y a partir de este punto el proceso es cíclico.

Se desea monitorear el nivel del tanque, la temperatura del agua, el estado de la válvula, la bomba y del elemento calefactor a través de un HMI. Además se desea controlar el nivel de llenado y vaciado del tanque así como la temperatura del agua.

3.3. Selección de hardware

Debido a que el proyecto implica representar un modelo de un control de proceso en la industria, se analiza el hardware del Industruino I/O así como de los transmisores de nivel y temperatura, para conocer las prestaciones que brinda cada uno de ellos.

3.3.1. Industruino I/O

Se seleccionó este tipo de controlador debido a que se ajusta a las necesidades requeridas para implementarlo en el proyecto, ya que cuenta con

canales para entradas/salidas analógicas (seleccionables por software) y digitales. Por esta razón se decidió optar por este tipo de controlador, que a diferencia de una placa tradicional de Arduino este cuenta con la capacidad de manejar voltajes superiores a los de las tarjetas convencionales lo que permite conectar con facilidad todo tipo de sensores y actuadores, además de que se pueden ingresar al equipo valores estandarizados de voltaje o corriente es decir de 0 a 10V o de 4 a 20mA de corriente continua. También cabe mencionar que se seleccionó este equipo ya que la topboard se basa en el estándar de la tarjeta Arduino Leonardo por lo que para su programación no se requiere de un software adicional sino que se puede utilizar el IDE de Arduino para poder programarlo.

3.3.1.1. Características técnicas del Industruino I/O

Industruino posee tres secciones: analógica, digital y del microcontrolador (MCU) que se encuentran aisladas para proteger el equipo.

Tabla 2

Características de la sección analógica

Sección analógica	
Canales de entrada analógicos	4
Canales de salida analógicos	2
Niveles analógicos	0-10V / 4-20 mA
Resolución de entrada analógica	18 bits
Resolución de salida analógica	12 bits

Fuente: (Industruino, 2016)

Tabla 3

Características de la sección digital

Sección digital	
Tensión de alimentación	6.5-32V
Canales digitales de E/S	8
Rango de tensión de entrada digital	3.3-32V

Continua

Rango digital de voltaje de salida	Igual a Vin (entrada 6.5-32V)
Corriente de salida por pin digital	2.3 A
Corriente total de salida	6 A

Fuente: (Industruino, 2016)

Tabla 4

Características de la sección MCU

Sección MCU	
Memoria Flash	32 KB de los cuales 4 KB utilizado por el gestor de arranque
EEPROM	4 KB
Velocidad de reloj	16 MHz
Tensión de funcionamiento del MCU	5V
Biblioteca LCD compatibles	U8G y UC1701

Fuente: (Industruino, 2016)

3.3.2. Transmisor de nivel ultrasónico

Para el control del nivel del líquido en el tanque, se seleccionó un transmisor de nivel ultrasónico modelo S18UIA con salida analógica de corriente estándar, que permite obtener una medición continua de la variable que se está controlando.

El transmisor ultrasónico se ajusta a las necesidades del proceso puesto que el fluido a controlar es agua que al ser enviada al tanque no genera espuma ni tampoco turbulencia. En la salida el transmisor presenta una señal analógica de 4 a 20 mA que puede ser adquirida fácilmente por el Industruino I/O mediante sus canales analógicos de entrada.

Tabla 5

Características de las series S18U con salida analógica

Especificaciones	
Rango sensible	30 a 300 mm
Voltaje de alimentación	10 a 30 Vdc
Frecuencia ultrasónica	300 Khz
Circuito de protección de suministro	Protegido contra polaridad inversa y tensiones transitorias
Configuración de salida	Salida analógica: 0 a 10 V dc o 4 a 20 mA dependiendo del modelo
Protección de salida	Protegido contra las condiciones de cortocircuito
Tiempo de respuesta de salida (Para un cambio de paso del 95%)	2.5 milisegundos: Cable negro a 5-30V dc 30 milisegundos: Cable negro a 0-2V dc (o abierto)
Retraso en la puesta en marcha	300 mili segundos
Efecto de la temperatura	0.02% de la distancia/°C
Linealidad	2,5 ms respuesta: ± 1 mm 30 ms respuesta: ± 0,5 mm
Resolución	2.5 ms respuesta: 1 mm 30 ms respuesta: 0.5 mm
Tamaño mínima de ventana	5 mm
Ajustes	Límites de la ventana de detección: La programación en modo TEACH de los límites de la ventana cercana y de la ventana lejana puede ajustarse mediante el pulsador o de forma remota a través de la entrada TEACH

Continua

Indicadores	Indicador de rango (rojo / verde)	Verde : El objetivo está dentro del rango de detección Rojo: El objetivo está fuera del alcance de detección OFF: La potencia de detección está desactivada
	Indicador de Enseñanza / Salida (Amarillo / Rojo)	Amarillo: El objetivo está dentro de los límites enseñados OFF: El objetivo está fuera de los límites de ventana enseñados Rojo: El sensor está en modo TEACH
Entrada TEACH remota	Impedancia:12KΩ	
Construcción	Barril roscado: Termoplástico poliéster Carcasa del botón pulsador: ABS / PC Tubos de luz: Acrílico	
Condiciones de operación	Temperatura: -20 ° a + 60 ° C (-4 ° a + 140 ° F) Humedad relativa máxima: 100%	
Conexiones	2 m (6,5 ') o 9 m (30') blindado de 5 conductores (con drenaje) PVC con encaje cable conectado o de 5 pines Desconexión rápida estilo Euro (consulte la página 10 para obtener opciones de cable de desconexión rápida)	
Clasificación Ambiental	El diseño a prueba de fugas está clasificado IEC IP67; NEMA 6P	

Continua

Vibración y choque mecánico	Todos los modelos satisfacen Mil. Std. 202F requisitos método 201A (vibración: 10 a 60 Hz máx., Doble amplitud 0.06 ", aceleración máxima 10G). También cumple con los requisitos IEC 947-5-2: 30G 11 ms duración, media onda senoidal.
Temperatura, calentamiento	Menos del 1,7% de la distancia de detección al encender
Certificaciones	CE cULus

Fuente:(BANNER, 2016)

3.3.3. Transmisor de temperatura

Para controlar la temperatura en el tanque, se escogió un transmisor de temperatura del tipo termistor, que permite obtener una medición continua de la variación de la temperatura. El transmisor presenta una gran sensibilidad es decir, una rápida variación a los cambios de temperatura, para su selección también se consideró que el agua no es abrasiva ni corrosiva por tanto no dañará el termopozo. El transmisor presenta en su salida una señal analógica estándar de voltaje de 0 a 5V cc que puede ser adquirida por el Industruino I/O utilizando los canales analógicos de entrada.

Tabla 6

Características del transmisor de temperatura

Transmisor de temperatura	
Sensor	Termistor
Tipo	Semiconductor 1uA/K
Voltaje mínimo de operación	5.7 V
Señal de salida	0 – 5 V cc
Estado	Funcional con un desplazamiento en la lectura.

Fuente: (Degem Systems, 2010)

3.4. Selección de elementos finales de control

Para el proyecto se requiere de una bomba centrifuga para enviar el fluido desde el depósito al tanque de proceso, un elemento calefactor para incrementar la temperatura del líquido y una válvula para drenar el agua del tanque. Estos elementos permitirán controlar el proceso mediante una secuencia que será programada en el Industruino I/O el cual permitirá que estos sean accionados según la lógica de programación.

3.4.1. Bomba centrifuga

Para su selección se tomó en cuenta la consistencia del fluido a transferir en este caso es líquido entonces se utilizara una bomba centrifuga, que tienen un gran desempeño a la hora de enviar agua de un lugar a otro. La distancia a la que se tiene que transportar el fluido no es considerablemente extensa por lo que no existe ningún inconveniente, además la bomba no estará permanentemente funcionando si no que lo hará únicamente cuando el proceso lo requiera.

Tabla 7

Características eléctricas de la bomba

Bomba centrifuga	
Voltaje nominal	12 V
Corriente de arranque	2.36 A
Corriente en estado estable a 12 V	2.33 A
Estado	Funcional

Fuente: (Degem Systems, 2010)

3.4.2. Resistencia calefactora

Se seleccionó la resistencia calefactora como elemento para calentar el agua en el tanque ya que las resistencias de inmersión permiten incrementar rápidamente la temperatura del fluido a través de transferencia de calor por contacto directo con el mismo.

Tabla 8

Características eléctricas de la resistencia calefactora

Elemento calefactor	
Voltaje nominal	220 Vrms
Resistencia	32 Ω
Estado	Funcional

Fuente: (Degem Systems, 2010)

3.4.3. Válvula de solenoide

Se seleccionó la válvula de solenoide debido a que, el caudal del fluido a controlar no genera mucha presión sobre esta. La cantidad de líquido en el tanque es de aproximadamente diez litros, es así que una válvula accionada eléctricamente puede realizar su función de abrir y cerrar sin ningún problema. La válvula es ideal para el proceso ya que su función es abrir o cerrar totalmente el paso del fluido hacia el deposito.

Tabla 9

Características eléctricas de la válvula de solenoide

Válvula solenoide	
Voltaje nominal	12 V
Corriente en estado estable	1 A
Estado	Funcional

Fuente: (Degem Systems, 2010)

3.4.4. Relé electromagnético

Se seleccionó el relé electromagnético de 12V/10A debido a que puede ser conectado directamente a las salidas del Industruino, que normalmente proporciona 12 V, para que funcione como un interruptor accionado eléctricamente para abrir o cerrar sus contactos dependiendo del estado de las salidas para accionar los elementos de control final.

Tabla 10

Características eléctricas del relé SRD-12VDC-SL-C

Especificaciones de la bobina	
Voltaje nominal	12V
Corriente nominal	70 a 90mA
Resistencia	55 a 70 ohmios
Potencia	0.45W
Especificaciones del contacto	
Capacidad para carga resistiva	28VDC@7A, 125VAC@10A, 240VAC@7A
Capacidad para carga inductiva	120VAC@5A, 28VDC@5A
Máximo voltaje	250VAC

Fuente: (Micro JPM S.A., 2017)

Una vez analizados los transmisores así como los elementos de control final que se requieren para ejecutar el proceso, se determinó que los instrumentos y elementos eléctricos que se encuentran en el equipo didáctico PCT-3 del laboratorio de instrumentación virtual, que consta de dos módulos: la consola de control (PCT-3/1) y el sistema físico (PCT-3/2), se ajustan a los requerimientos necesarios para desarrollar el proceso de llenado y vaciado de un tanque. El equipo didáctico consta de dos procesos: uno de nivel y otro de temperatura donde el transmisor de nivel proporciona una señal de salida estándar de corriente de 4 – 20 mA y el transmisor de temperatura proporciona una señal de salida de 0 – 5 Vcc que serán adquiridas por el Industruino para controlar el proceso.

3.5. Selección de software

Para desarrollar la programación del Industruino I/O se utiliza el IDE de Arduino y para realizar la HMI se utiliza el software de LabVIEW.

3.5.1. IDE de Arduino

La razón por la que se utiliza el IDE de Arduino es debido a que el Industruino I/O es un controlador lógico versión Arduino, es así que su software es totalmente compatible con el entorno de desarrollo integrado de

Arduino, por lo que basta con conectar el equipo a la PC mediante el cable micro USB para que se instalen los drivers del Industruino y se cree automáticamente un puerto COM en la PC, de esta manera al abrir el software de Arduino el equipo pueda ser reconocido como una placa Arduino Leonardo.

3.5.2. LabVIEW

En cuanto al software para desarrollar la HMI se utiliza LabVIEW para crear instrumentos virtuales que ayudan a visualizar de mejor manera el estado del proceso que se lleva a cabo. Se eligió este software porque permite establecer una comunicación entre la PC y el Industruino para enviar y recibir datos de manera serial para controlar el proceso

3.6. P&ID del proceso

El diagrama de instrumentación y tubería se realiza en base al proceso que se va a controlar en este se muestra los símbolos de los instrumentos así como de los elementos de control que se utilizan en el proyecto.

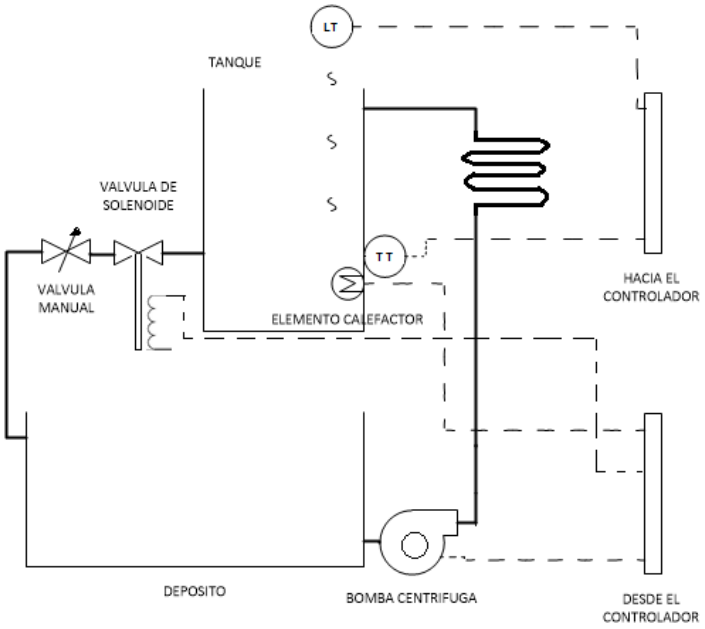


Figura 32 P&ID del proceso de llenado y vaciado.

Fuente: (Mallitasig, 2016)

3.7. Diagrama de flujo de la programación.

El diagrama de flujo de la programación representa los pasos a seguir para realizar los algoritmos de programación tanto para el Industrio así como para la HMI en base al proceso a controlar.

Las letras (a, b, c, d, e, f, p) representan los datos que recibe el Industruino para que pueda ejecutar las instrucciones según la letra correspondiente. Mientras que los valores máximos y mínimos de nivel y temperatura se utilizan para comparar, en LabVIEW, con los valores provenientes desde el Industruino.

Donde:

La letra (a) representa el encendido de la bomba.

La letra (b) representa el apagado de la bomba.

La letra (c) representa el encendido del elemento calefactor.

La letra (d) representa el apagado del elemento calefactor.

La letra (e) representa el encendido de la electroválvula.

La letra (f) representa el apagado de la electroválvula.

La letra (p) representa el apagado de la bomba, del elemento calefactor y de la electroválvula.

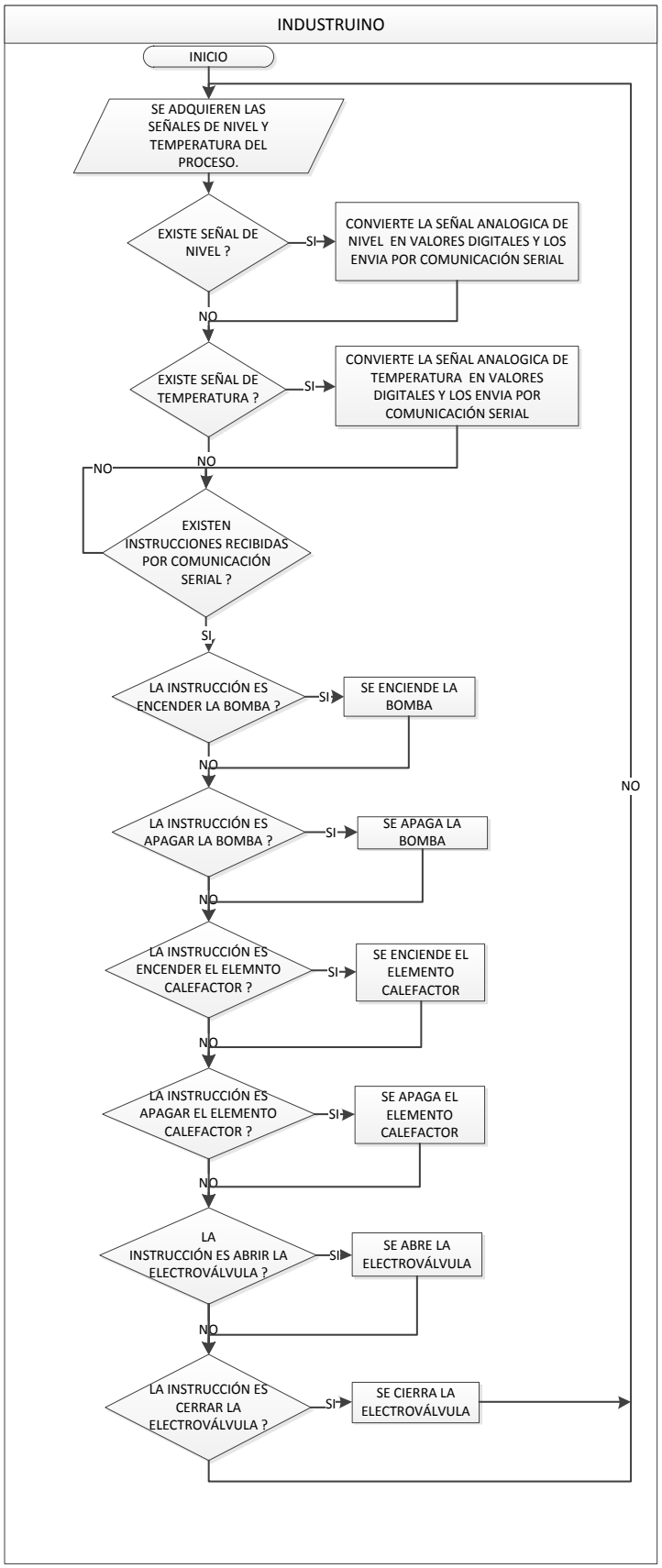


Figura 33 Diagrama de flujo de la programación del IND.

Fuente: (Mallitasig, 2016)

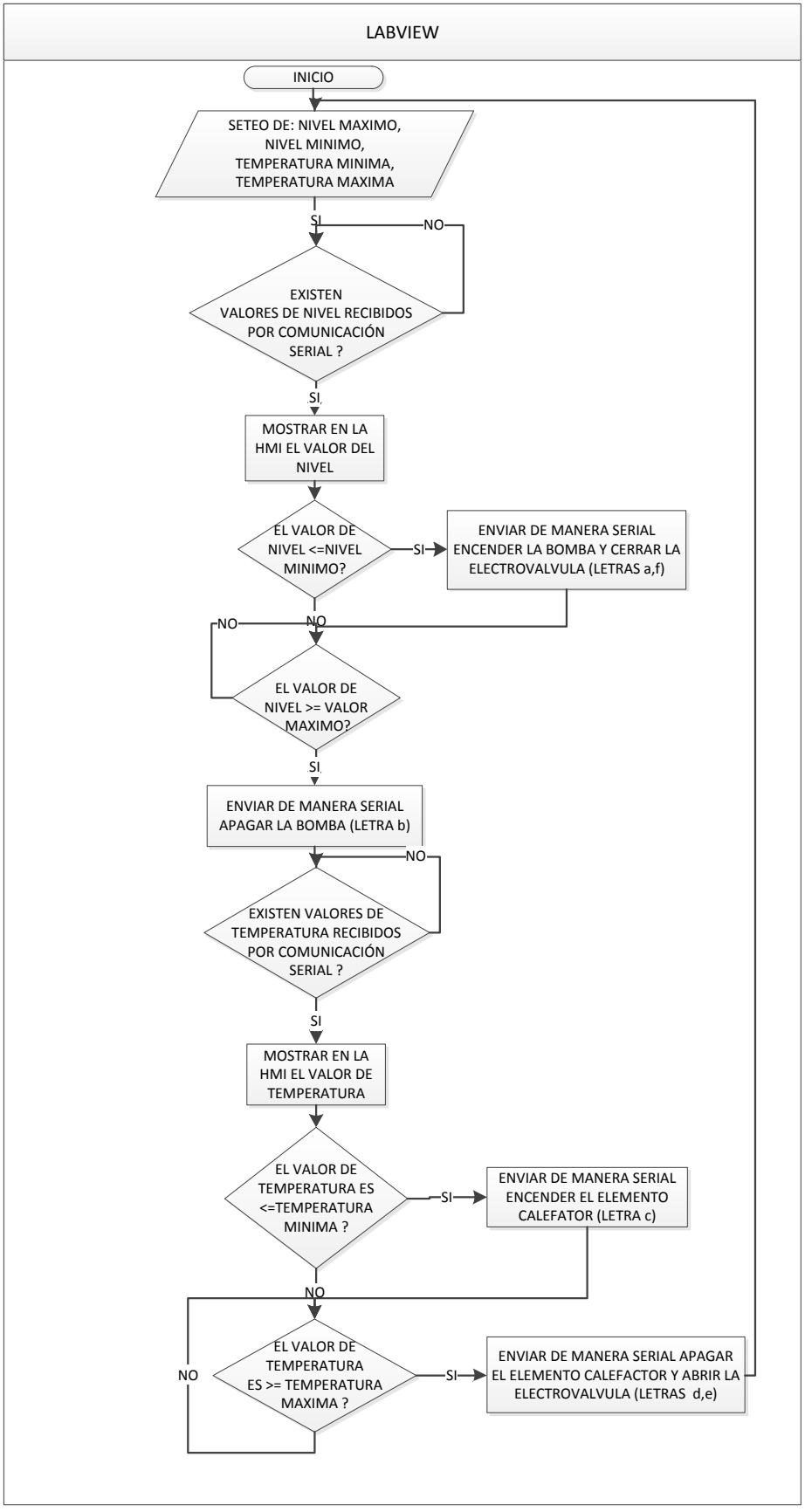


Figura 34 Diagrama de flujo de la programación de LabVIEW

Fuente: (Mallitasig, 2016)

3.8. Montaje del Industruino I/O

El montaje del Industruino se realiza, previo a la instalación de los drives en la PC, en base a la guía de montaje donde se detalla paso a paso como se debe ensamblar el equipo es decir acoplar mediante los conectores FPC la topboard con la placa base y con los botones de membrana en las ranuras correspondientes sin doblar los cables para esto se requiere únicamente de un destornillador estrella pequeño.

3.9. Instalación de los drivers del Industruino

Los drivers se instalan automáticamente cuando se conecta el Industruino I/O a la PC mediante el cable micro USB, estos drivers crean un puerto COM que permitirá reconocer al equipo como una placa Arduino Leonardo y posteriormente mediante el mismo puerto cargar el código de programación desde el IDE de Arduino además de establecer una comunicación serial entre ambos dispositivos.

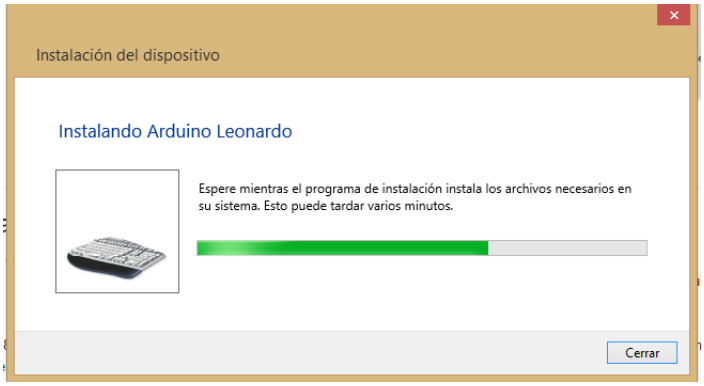


Figura 35 Instalación del Industruino a la PC

Fuente: (Mallitasig, 2016)

3.10. Archivos de soporte de Industruino

Para poner en marcha el Industruino I/O se requiere de los archivos de soporte en este caso el paquete de archivos contiene las librerías de EthernetIndustruino, Indio y UC1701 así como ejemplos de cada una de ellas, estas librerías soportan versiones de Arduino 1.6.5 en adelante por lo que no existe inconveniente ya que para nuestro proyecto se utiliza la versión 1.6.7.

3.11. Instalación de las librerías Indio y UC1701

La instalación de la librería Indio en el software de Arduino ayuda a que se pueda utilizar el mismo lenguaje de programación que se utiliza para programar las tarjetas convencionales de Arduino, por otro lado la librería UC1701 permite utilizar la pantalla del Industruino como un LCD normal. A continuación se describen el procedimiento a seguir para instalar las librerías.

Descargar el paquete de archivos de soporte para el Industrio, se lo puede bajar directamente de la página oficial de Industruino. Una vez descargado el paquete de archivos que contienen las librerías, proceda a instalarlas en su computadora. Luego localizar el archivo descargado en su computadora, copiar el archivo a una nueva carpeta donde desee extraerlo, dar clic derecho sobre el archivo y luego clic en extraer aquí.

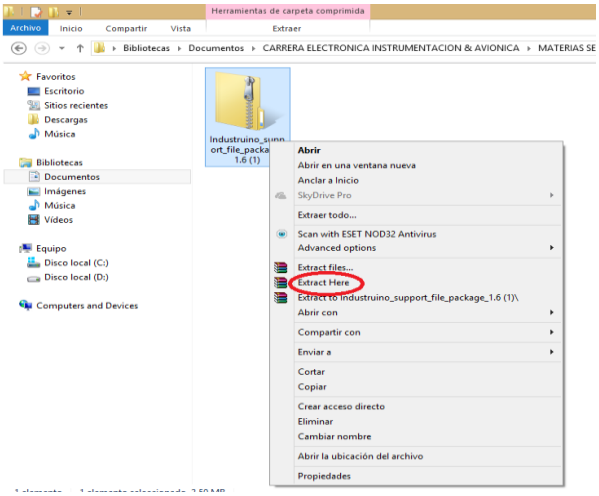


Figura 36 Archivo Industruino support file package 1.6

Fuente: (Mallitasig, 2016)

Se extraerán dos archivos, las librerías que necesita se encuentran en el archivo: Industruino support file package 1.6, al abrirlo aparece un archivo (Libraries), donde están archivos como UC1701 para el LCD y el archivo Indio para el Industruino, en cada una de estas se encuentran las librerías y ejemplos de código para empezar a utilizar el Industruino.

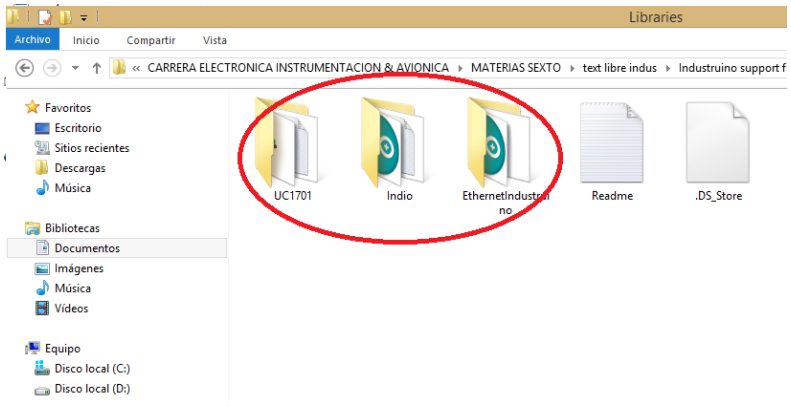


Figura 37 Librerías Indio, UC1701.

Fuente: (Mallitasig, 2016)

Una vez que tenga localizadas las librerías, proceda a cargarlas en el Software de Arduino para que se incluyan con las demás librerías de Arduino. Para esto abra el IDE de Arduino, en la parte superior dar clic en la pestaña programa y seleccionar incluir librería luego clic en añadir librería .ZIP.

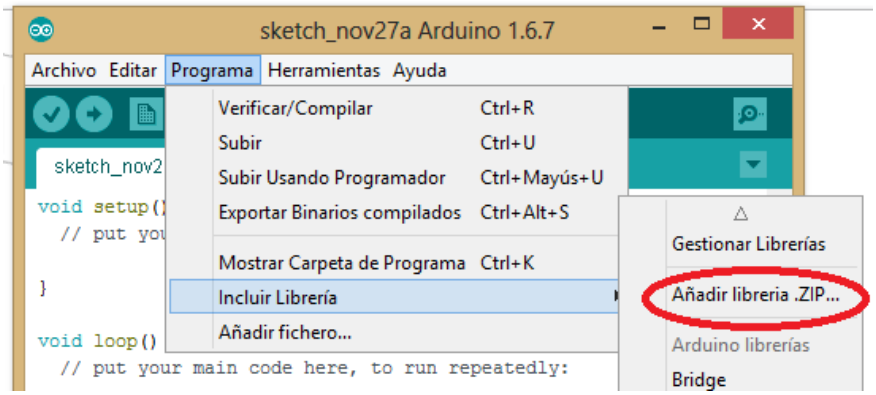


Figura 38 Añadir librería .ZIP en el software Arduino

Fuente: (Mallitasig, 2016)

Encontrar la ubicación de los archivos que contienen las librerías, seleccionar la carpeta que contiene la librería que se desea añadir en este caso la librería Indio. A continuación dar clic en abrir.

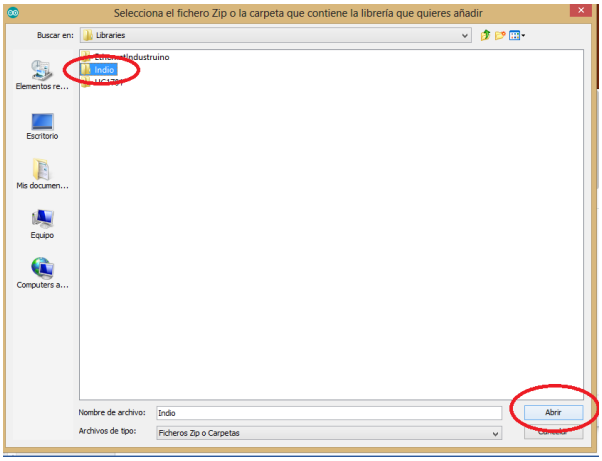


Figura 39 Selección de la carpeta que contiene la librería.

Fuente: (Mallitasig, 2016)

Para comprobar que la librería se incluyó correctamente se ingresa nuevamente en la pestaña programa, seleccionar incluir librería y aparecerá una lista de todas las librerías que se encuentran instaladas, verificar que se encuentre la librería que se instaló en la parte inferior.

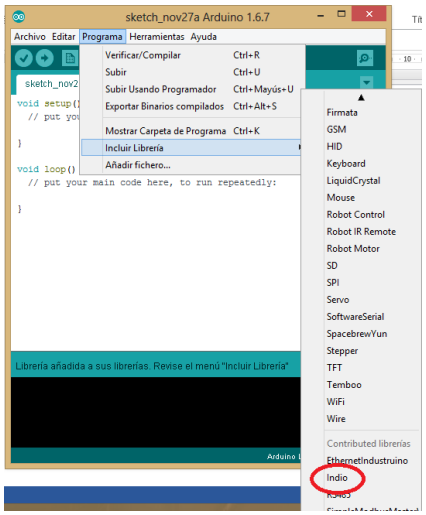


Figura 40 Librería incluida en el software Arduino

Fuente: (Mallitasig, 2016)

Una vez que se ha verificado que la librería está incluida en el software, realice el mismo procedimiento anterior para cargar la librería UC1701 para utilizar el LCD y de igual manera comprobar que esté incluida con las demás librerías. Cuando se ha terminado las instalaciones ya se puede utilizar los

bocetos que incluyen las librerías como ejemplos para desarrollar la programación del proyecto.

3.12. Configuración del transmisor ultrasónico

La configuración del transmisor se realiza mediante un único botón (TEACH) que se encuentra ubicado en el transmisor el cual permite enseñar al transmisor a reconocer dos límites uno mínimo y otro máximo.

Además el transmisor incorpora dos leds (Power ON/OF led y Output/Teach led) que sirven como indicadores para realizar la configuración, a continuación se detalla la indicación de cada uno.

Tabla 11

Estado del led PWR

Led Power	Indicación
Apagado	La alimentación esta desactivada
Encendido en rojo	El objetivo es débil o está fuera del rango de detección.
Encendido en verde	El sensor está funcionando normalmente, buen objetivo.

Fuente: (BANNER, 2016)

Tabla 12

Estado del led OUT

Led Output / Teach	Indicación
Apagado	El objetivo está fuera de los límites de la ventana
Amarillo	El objetivo está dentro de los límites de la ventana
Encendido en rojo (solido)	En modo de enseñanza, esperando el primer límite.
Encendido en rojo (parpadeando)	En modo de enseñanza, esperando el segundo límite.

Fuente: (BANNER, 2016)

Para empezar la configuración del transmisor primero se debe identificar los cables del transmisor (Ver figura 43), en las hojas técnicas, que se van a utilizar, para ello basta con observar sus colores para reconocerlos y determinar cuál es su función, de esta manera los cables marrón y azul (bn, bu) son la alimentación del transmisor, para este caso esta alimentado con 15 V dc mediante la consola de control PCT-3/1, el cable blanco (wh) es la salida de 4 -20 mA, en donde se coloca un amperímetro en serie para verificar el valor de salida al momento de realizar la programación.

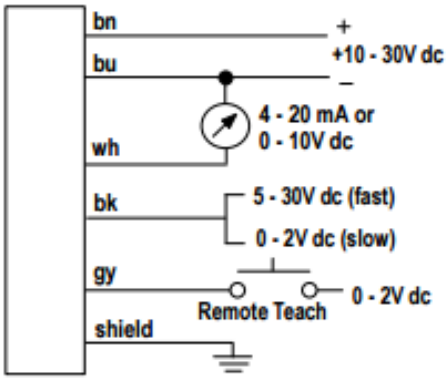


Figura 41 Diagrama de cables del transmisor

Fuente: (BANNER, 2016)

Una vez que se ha identificado el estado de cada uno de los leds y los cables del transmisor, se procede a configurarlo mediante el pulsador TEACH, a continuación se indica en las tablas el procedimiento a seguir.

Tabla 13

Entrar al modo de configuración

Método	Acción	Resultado
Presionar el botón	Mantenga presionado el botón TEACH 	Led OUT: encendido en rojo Led PWR : Encendido en verde (buena señal), encendido en rojo (sin señal)

Fuente: (BANNER, 2016)

A continuación presentar el primer límite es decir el nivel mínimo (en este caso se establece el nivel mínimo en nueve centímetros). Para esto el led PWR debe estar encendido en verde.

Tabla 14

Configurar el primer límite (nivel mínimo)

Método	Acción	Resultado
Presionar el botón	Presione el botón TEACH una vez. 	Enseñar aceptado: El sensor aprende el límite de 4 mA. Led OUT: parpadeando en rojo Enseñar no aceptado: Led OUT: encendido en rojo

Fuente: (BANNER, 2016)

Luego se presenta el segundo límite es decir el nivel máximo (en este caso se establece el nivel máximo en veinte centímetros). El led PWR debe estar encendido en verde.

Tabla 15

Configurar el segundo límite (nivel máximo)

Método	Acción	Resultado
Presionar el botón	Presione el botón TEACH una vez. 	Enseñar aceptado: El sensor aprende el límite de 20 mA. Led OUT: amarillo o apagado Enseñar no aceptado: Led OUT: parpadeando en rojo

Fuente: (BANNER, 2016)

3.13. Calibración analógica del Industruino I/O

La calibración del Industruino se realiza para aumentar la precisión de las entradas analógicas y porque se tiene inconvenientes para adquirir la señal

de corriente. Para este caso se va actualizar el bloque correspondiente al ADC de corriente de 4 – 20 mA. A continuación se describe el procedimiento.

Primero se conecta el Industruino a una fuente de alimentación externa, y a la PC mediante el cable micro USB.

A continuación se carga el código de programación que contiene los datos para la calibración usando el IDE de Arduino (ver anexo D).

Luego se abre el monitor de serie para visualizar las opciones que ejecuta el programa, en este caso se elige la opción número cuatro, a continuación se debe ir ingresando y anotando los valores solicitados por el programa, los valores deben estar en microamperios.

La calibración empieza con valores bajos de corriente en este caso se requiere que el rango sea desde 4 mA, por lo que se conecta dicho valor de corriente a la entrada del canal uno, se utiliza un amperímetro para establecer el valor en microamperios, a continuación se debe anotar el valor de la corriente y el valor (raw) correspondiente que muestre el monitor serie, esto se realiza para cada canal.

Tabla 16

Valores bajos de corriente en la calibración

Canal	uA	raw
CH1	3090	263
CH2	3089	248
CH3	3089	245
CH4	3090	259

Tabla 17

Valores altos de corriente en la calibración

Canal	uA	raw
CH1	20000	1232
CH2	20000	1237
CH3	20000	1236
CH4	20000	1231

Para finalizar se abre el archivo Indio.cpp y se reemplazan los valores existentes por los valores obtenidos y se guarda los cambios realizados.

3.14. Desarrollo del código de programación

Una vez que se tiene las librerías incluidas en el software de Arduino así como programado el transmisor ultrasónico, se procede a desarrollar el código de programación para controlar el proceso, para esto se conectan los dos transmisores en los canales analógicos de entrada es decir en CH1 para adquirir la señal analógica del nivel y en CH2 para adquirir la señal analógica de temperatura, este último transmisor no requiere de ninguna configuración. A continuación se muestra la conexión.

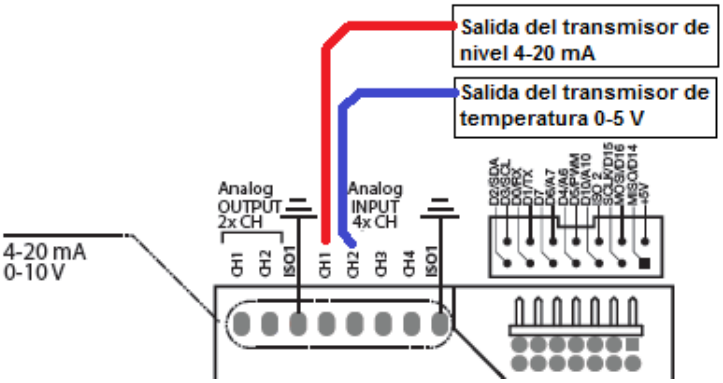


Figura 42 Conexión de los transmisores a las entradas analógicas

Fuente: (Mallitasig, 2016)

Para este caso se enviaran letras desde el monitor serie de Arduino para que el Industruino las reconozca y ejecute las instrucciones correspondientes.

En primera instancia se utiliza el monitor serie de Arduino para comprobar el funcionamiento del programa, posteriormente una vez que se termine el código de programación se va a desarrollar una HMI en el software de LabVIEW para crear una interfaz de comunicación serial entre el Industruino y la PC.

3.14.1. Declaración de librerías

La librería que se incluyen en el código de programación es la de Indio, adicionalmente al incluir dicha librería automáticamente también se incluye la librería Wire, ambas son necesarias para programar el Industruino.

```
#include <Indio.h>

#include <Wire.h>
```

3.14.2. Declaración de variables

En este caso se utiliza una única variable que permite almacenar las letras enviadas desde el monitor serie y guardarlas en una variable de tipo char (caracter).

```
char dato;
```

3.14.3. Inicialización del programa

Empieza con la siguiente línea de código que se escribe a continuación, dentro la estructura void setup se coloca el código que se ejecutara una sola vez cuando se energice el Industruino.

```
void setup()

{
```

3.14.4. Configuración de la comunicación serial

Para establecer la comunicación entre ambos dispositivos (Industruino I/O y PC) se utiliza la comunicación serial para enviar y recibir datos a una velocidad determinada, para esto se utiliza la siguiente línea de código la cual abre el puerto de comunicación a una velocidad de 9600 baudios.

```
Serial.begin (9600);
```

3.14.5. Configuración de la resolución del ADC

Para este caso que se adquiere una señal analógica de corriente y una de voltaje, el Industruino posee internamente un convertidor analógico digital ADC, que convierte las señales analógicas en valores digitales en un rango determinado conocido como resolución, el mismo que está fijado a una resolución de 12 bits es decir para un rango comprendido entre 0 y 4095.

```
Indio.setADCResolution(12);
```

3.14.6. Configuración del modo las entradas analógicas

En este caso se utiliza el canal 1 (CH1) para ingresar la señal analógica de corriente y el canal 2 (CH2) para ingresar la señal analógica de voltaje como en este proyecto se necesita enviar los valores de las lecturas y mostrarlos en el monitor serie, se fija el modo de lectura a corriente (1, mA_raw) y a voltaje (2, V10_raw) de esta manera los valor estarán comprendidos en el rango de 0 – 4095 para ambas señales.

```
Indio.analogReadMode(1,mA_raw);

Indio.analogReadMode(2,V10_raw);
```

3.14.7. Configuración del modo de las salidas analógicas

El CH1 envía un valor de 12V para activar y 0V para desactivar la bobina del relé que controla el encendido y apagado del elemento calefactor.

El CH2 envía un valor de 12V para activar y 0V para desactivar la bobina del relé que controla el encendido y apagado de la electroválvula. Para esto se utilizan las siguientes líneas de código.

```
Indio.analogWriteMode(1, V10);

Indio.analogWriteMode(2, V10);
```

Además para que los relés no se accionen al energizar el Industruino se utilizan las siguientes líneas de código, donde 0 equivale a cero voltios y false significa que no almacena ese valor en la EEPROM del DAC.

```
Indio.analogWrite(1,0,false);

Indio.analogWrite(2,0,false);
```

3.14.8. Configuración de la salida digital

El CH1 envía un valor de 12V para activar y 0V para desactivar la bobina del relé que controla el encendido y apagado de la bomba centrífuga. En la programación se designó éste canal en modo salida (OUTPUT). Para esto se utilizan las siguientes líneas de código.

```
Indio.digitalMode(1,OUTPUT);
```

Para que el relé no se accione al energizar el Industruino se utiliza la siguiente línea de código.

```
Indio.digitalWrite(1,LOW);  }
```

3.14.9. Estructura del control

Empieza con la línea de código que se escribe a continuación, dentro de la estructura void loop se coloca la programación principal así como las instrucciones que se van a estar ejecutándose de manera cíclica mientras el Industruino esté conectado a una fuente de alimentación.

```
void loop()

{
```

3.14.10. Lectura de las variables de nivel y temperatura.

Para leer las señales de nivel y temperatura provenientes de los transmisores se utiliza las siguientes líneas de código, que a su vez el valor que se obtiene se lo almacena en una variable tipo string.

```
String CH1=String(Indio.analogRead(1),DEC);    // Lee el nivel

String CH2=String(Indio.analogRead(2),DEC);  // Lee la temperatura
```

3.14.11. Envió de datos de manera serial

Debido a que se tiene que enviar dos valores al mismo tiempo, para poder identificarlos en el monitor serie, se envía cada valor junto con una letra y se los almacena en una nueva variable tipo string.

```
String datos1=String("A"+CH1);

String datos2=String("B"+CH2);

Serial.print(datos1);

Serial.print(datos2);
```

3.14.12. Comunicación serial

Las siguientes líneas de código permiten saber si la comunicación está disponible, además de leer y almacenar el dato recibido en una variable para ser utilizada más adelante.

```
if(Serial.available()>0)  {
 dato=Serial.read();
```

3.14.13. Control de encendido/apagado de la bomba

Las siguientes líneas de código permiten activar el relé que enciendo o apaga la bomba centrifuga según el dato recibido.

```
if (dato=='a')
{
Indio.digitalWrite(1,HIGH);
}
if (dato=='b')
{
Indio.digitalWrite(1,LOW);
 }
```

3.14.14. Control de encendido/apagado del elemento calefactor

Las siguientes líneas de código permiten activar el relé que enciendo o apaga el elemento calefactor de acuerdo al dato recibido.

```
if (dato=='c')
{
Indio.analogWrite(1,  10,false);
}
if (dato=='d')
{
Indio.analogWrite(1,  0,false);
}
```

3.14.15. Control de encendido/apagado de la electroválvula

Las siguientes líneas de código permiten activar el relé que abre o cierra la electroválvula según el dato recibido.

```
if (dato=='e')
{
Indio.analogWrite(2,  10,false);
```

```
}
if (dato=='f')
{
Indio.analogWrite(2,  0,false);
}
```

3.14.16. Subrutina paro

Las siguientes líneas de código permiten re direccionar es decir, salir del programa principal para ejecutar ciertas instrucciones en la subrutina "paro".

```
if (dato=='p')
{
paro();
}
}
}
```

Dentro de la subrutina paro se encuentran las instrucciones que el Industruino debe ejecutar cuando el programa ha sido re direccionado a esta subrutina es decir, aquí el programa deja de energizar la bobinas de los relés para detener el proceso. A continuación se muestran las líneas de código que realizan estas instrucciones.

```
void paro()
{
Indio.digitalWrite(1,LOW);
Indio.analogWrite(2,  0,false);
Indio.analogWrite(1,  0,false);
}
```

Compilamos el programa en busca de errores, para luego subir el código del programa al Industruino I/O.

3.15. Programación en LabVIEW

Una vez que se tiene el código de programación del Industruino, se procede a diseñar la HMI en el panel frontal de LabVIEW para luego desarrollar la programación en el diagrama de bloques.

3.15.1. Diseño de la HMI en el panel frontal

Para empezar colocamos los elementos necesarios en el panel frontal, los cuales representan los instrumentos virtuales que permiten monitorear el proceso.

Colocar un tanque y un termómetro los cuales permitirán visualizar el nivel y la temperatura del proceso, estos indicadores numéricos se encuentran en la paleta de controles.

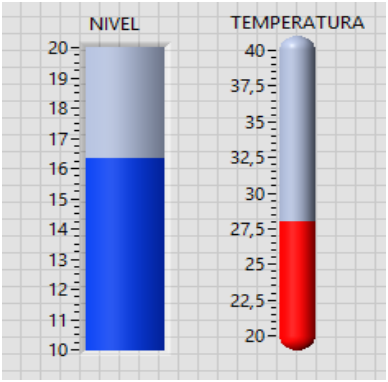


Figura 43 Indicadores de nivel y temperatura

Fuente: (Mallitasig, 2016)

A continuación colocar tres controles numéricos de desplazamiento vertical que se encuentran en la paleta de controles, para representar los selectores de nivel máximo (20 cm), nivel mínimo (10 cm se fija desde diez y no desde los nueve centímetros con la finalidad de evitar que cuando se fije el nivel mínimo no se pierda la lectura del transmisor) y temperatura mínima () máxima.

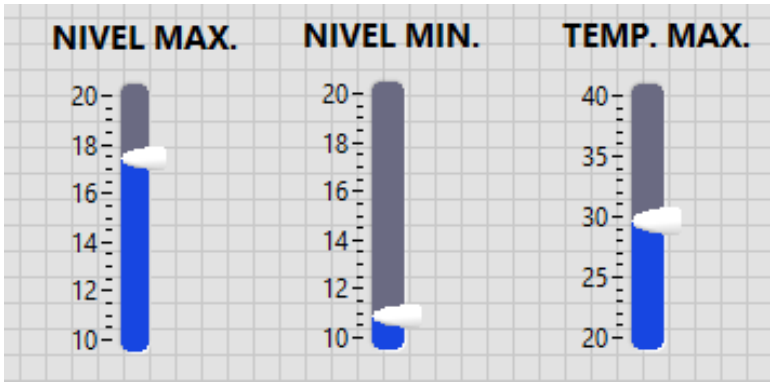


Figura 44 Controles de nivel y temperatura

Fuente: (Mallitasig, 2016)

Luego colocar cuatro ledos que se encuentran en la paleta de controles, para representar las alarmas de los valores máximos y mínimos de las variables de nivel y temperatura.

Figura 45 Indicadores de alerta de nivel y temperatura

Fuente: (Mallitasig, 2016)

Además se coloca indicadores para la bomba y para la electroválvula que se encuentran en la paleta de controles sección módulo DSC controles 3D y para el elemento calefactor se coloca un led indicador.

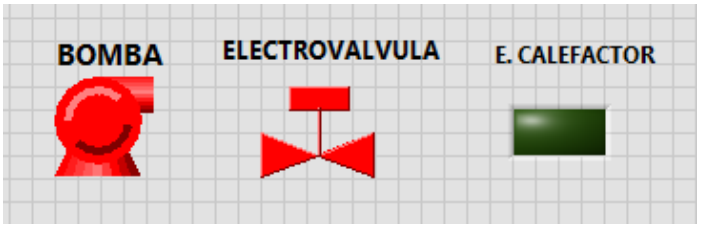


Figura 46 Indicadores para los elementos de control final

Fuente: (Mallitasig, 2016)

Para finalizar se coloca un selector de puertos para seleccionar el puertos COM cuando se conecte el Industruino a la PC este selector aparece automáticamente al colocar el VI configuración del puerto serial en el diagrama de bloques. También colocamos un botón booleano para utilizarlo como paro de emergencia del proceso.

Figura 47 Paro de emergencia y selector de puerto COM

Fuente: (Mallitasig, 2016)

Una vez que se tiene todos los controles y los indicadores se procede a colocar todo de una manera ordenada, además se coloca unos tanques así como tubería para interconectar el proceso. Para finalizar la HMI se coloca avisos para manipular los controles.

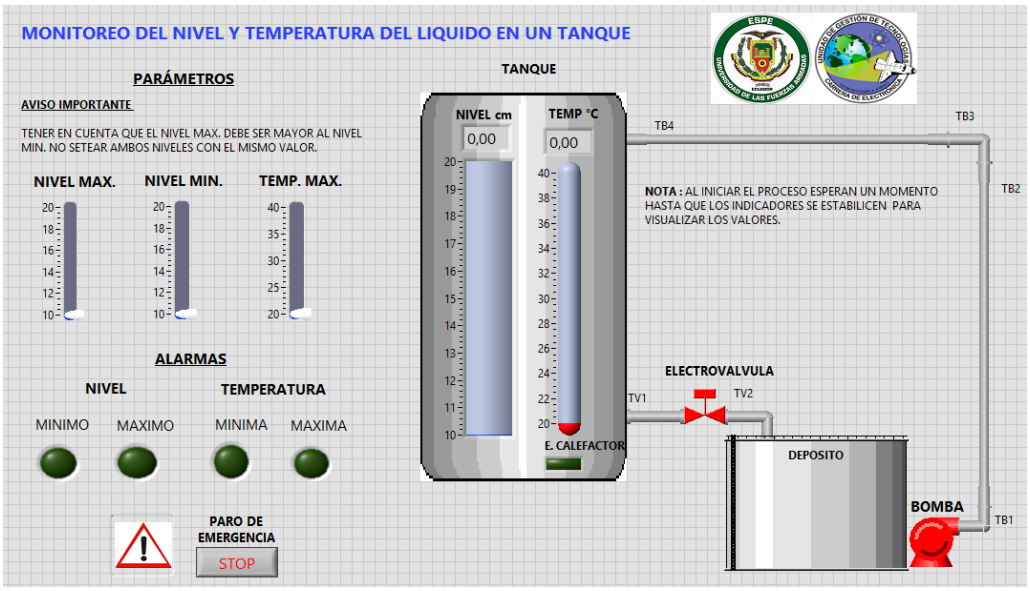


Figura 48 HMI para monitorear el nivel y temperatura en un tanque

Fuente: (Mallitasig, 2016)

3.15.2. Programación en el diagrama de bloques

En el diagrama de bloques se realiza la programación que se utiliza para interactuar con los elementos colocados en el panel frontal, además se realizan las conexiones necesarias entre los controles y los indicadores junto con otros elementos de la paleta de funciones, para representar los valores enviados desde el Industruino así como para poder enviar caracteres al equipo para que ejecute las instrucciones necesarias para el control.

Para establecer la comunicación serial entre el software LabVIEW y el Industruino primero se realiza a configuración del puerto serial, para lo cual se coloca el icono VISA Configure Serial Port que se encuentra en la sección Instruments I/O de la paleta de funciones. Aquí se coloca un control para seleccionar el puerto COM así como también se establece la velocidad de transmisión y la cantidad de bits dentro de constantes numéricas.

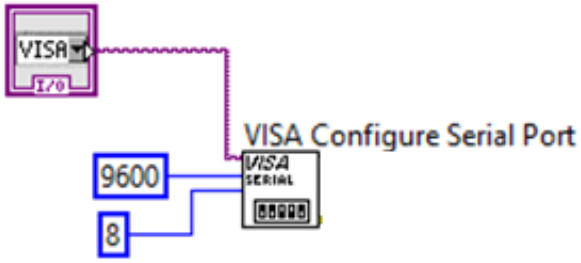


Figura 49 Configuración del puerto serial

Fuente: (Mallitasig, 2016)

Para poder leer los datos que ingresan a través del puerto serial se procede a configurar la lectura, para esto se utiliza VISA Read, se coloca un Property Node el cual se conecta al nodo de byte count del VISA Read para determinar el tamaño de bits que ingresan a través del puerto, para seleccionar esto dar clic sobre Serial Settings: Number of Bytes at Serial Port, además se utiliza una estructura de secuencia para colocar un tiempo de lectura entre los datos. Para leer los datos de nivel se utiliza un tiempo de 50 ms mientras que para la temperatura se establece un tiempo de lectura de 100 ms esto con la finalidad de poder obtener los dos valores sin que exista perdida de datos debido a que ambos valores son enviados a través del mismo puerto y al mismo tiempo

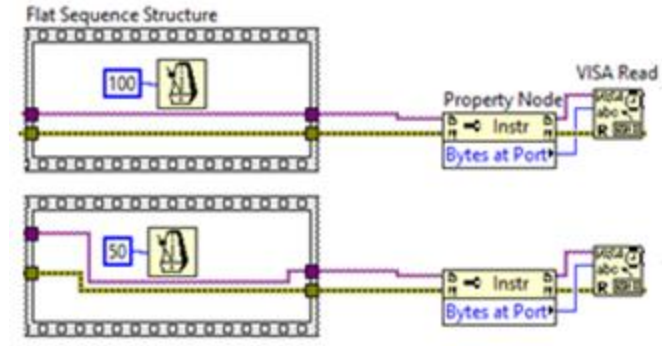


Figura 50 Configuración del VISA Read

Fuente: (Mallitasig, 2016)

Para procesar los datos recibidos se utiliza un Match Pattern el cual toma el dato de tipo String para luego separar el valor numérico de la constante (letra A) y presentar a su salida únicamente el valor numérico. Debido a que los datos de nivel y temperatura se envían al mismo tiempo para identificarlos se coloca al inicio de cada valor una constante tipo String y al ingresar al Match Pattern este tiene ya definida una letra para seleccionar el dato que le corresponde, así para el nivel se eligió la letra A y para la temperatura la letra B. Luego el dato que se tiene en la salida (after substring) se procede a convertirlo en un caracter numérico, para esto se utiliza Decimal String to Number. Se realiza el mismo procedimiento para el dato de nivel y de temperatura.

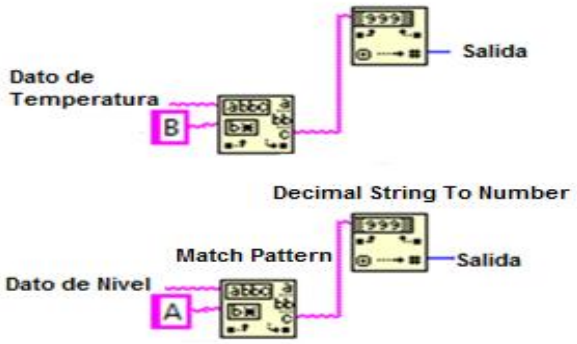


Figura 51 Conversión de decimal string a numérico

Fuente: (Mallitasig, 2016)

Una vez que se tienen los valores en tipo numérico (valores del ADC) se procede a realizar el acondicionamiento de señal para representarlos en valores de nivel y temperatura.

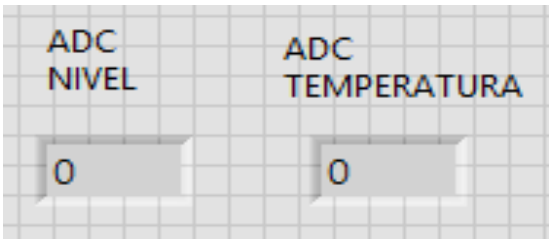


Figura 52 Nivel y temperatura en valores del ADC

Fuente: (Mallitasig, 2016)

Para realizar el acondicionamiento se señal del nivel se coloca el nivel del líquido del tanque en el nivel máximo fijado y se anota el valor que muestre el indicador del ADC de nivel en el panel frontal, luego bajar el nivel del líquido hasta el nivel mínimo fijado y anotar el valor.

Tabla 18

Valores para acondicionar la señal de nivel.

Magnitud	Valor mínimo	Valor máximo
Corriente	4 mA	20 mA
Nivel	9 cm	20cm
ADC	710	3677

A continuación se procede a realizar el acondicionamiento de señal utilizando la ecuación de la pendiente.

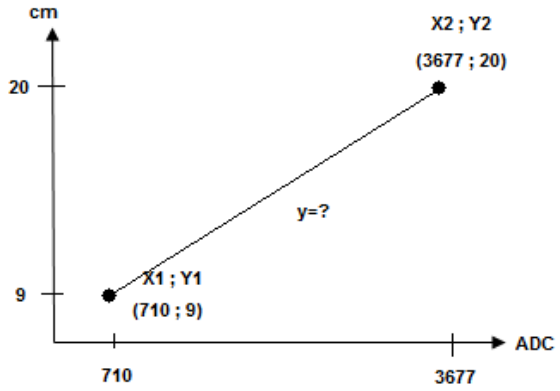


Figura 53 Representación del nivel en valores del ADC

Fuente: (Mallitasig, 2016)

$$m = \frac{Y2 - Y1}{X2 - X1} = \frac{20 - 9}{3677 - 710} = \frac{11}{2967} = 0{,}003707448$$

$$y = m(X - X1) + Y1 = 0{,}003707448(X - 710) + 9$$

Para realizar el acondicionamiento de la señal de temperatura se enciende el elemento calefactor para subir la temperatura del líquido hasta alcanzar los 20° C que es valor mínimo y se anota el valor que indica el ADC de temperatura de igual manera se anota el valor que muestre el indicador cuando la temperatura alcance los 40°C con estos valores se realiza el acondicionamiento.

Tabla 19

Valores para acondicionar la señal de temperatura

Magnitud	Valor mínimo	Valor máximo
Voltaje	0 V	2 V
Temperatura	20 °C	40 °C
ADC	19	747

A continuación se procede a realizar el acondicionamiento de señal utilizando la ecuación de la pendiente.

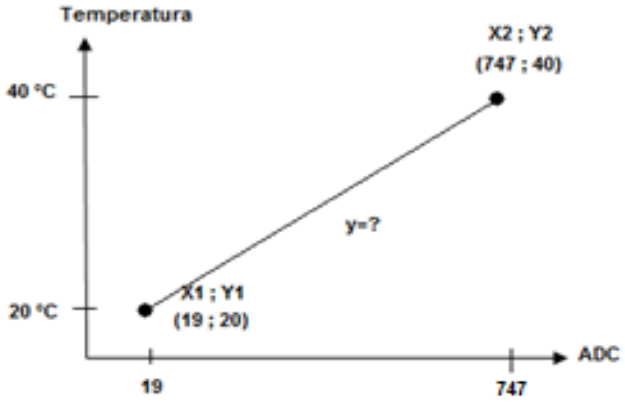


Figura 54 Temperatura en valores del ADC

Fuente: (Mallitasig, 2016)

$$m = \frac{Y2 - Y1}{X2 - X1} = \frac{40 - 20}{747 - 19} = \frac{20}{728} = 0{,}02747252747$$

$$y = m(X - X1) + Y1 = 0{,}02747252747(X - 19) + 20$$

Luego se colocan los valores obtenidos de las ecuaciones en el diagrama de bloques utilizando funciones numéricas para poder obtener el valor del nivel y la temperatura.

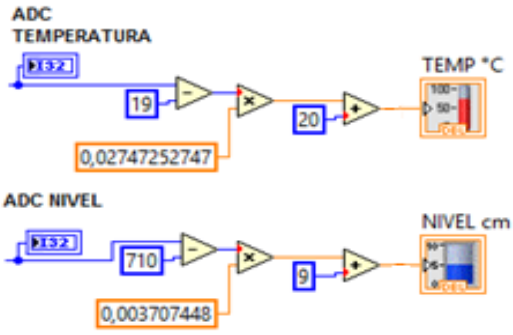


Figura 55 Acondicionamiento de las señales

Fuente: (Mallitasig, 2016)

Una vez que se tienen los valores de nivel y temperatura se procede a realizar la comparación de estos con valores establecidos en los controles numéricos que se colocaron en el panel frontal es decir en la HMI.

Primero se realiza la comparación entre valores para el nivel, en este caso entre el nivel máximo y en nivel mínimo, lo cual permitirá el encendido o apagado de la bomba, para esto se utiliza un biestable RS mediante

compuertas NOR en el diagrama de bloques, donde su función es almacenar un estado lógico de acuerdo al estado en sus entradas.

De igual manera se realiza la comparación entre valores para la temperatura entre la temperatura máxima y la temperatura mínima, para encender o apagar el elemento calefactor, para este caso también se utiliza un biestable mediante compuertas NOR.

También se realiza la comparación entre el nivel mínimo y la temperatura máxima para abrir o cerrar la electroválvula, de igual manera se utiliza un biestable mediante compuertas NOR.

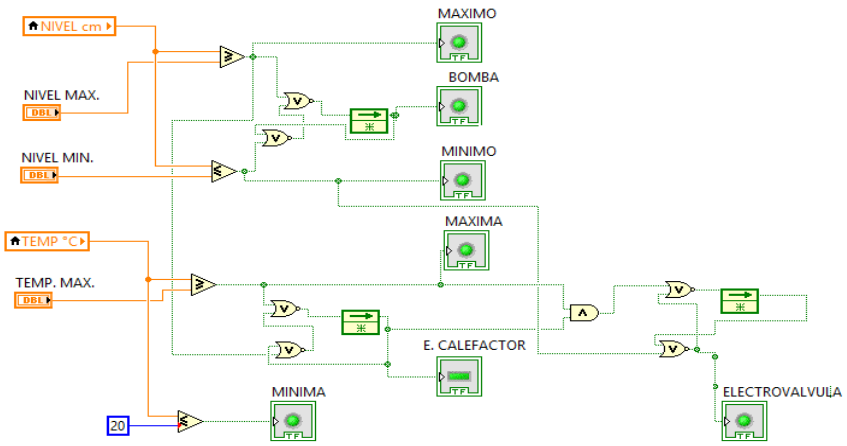


Figura 56 Comparación entre los valores de nivel y temperatura

Fuente: (Mallitasig, 2016)

Para enviar las órdenes a los elementos de control final se utiliza una estructura de casos en el diagrama de bloques, para comparar el estado de los indicadores según los valores de nivel y temperatura, a la entrada de la estructura y enviar la letra correspondiente según sea el caso es decir verdadero o falso, para esto se utilizan tres estructuras de casos, para la bomba, la electroválvula y el elemento calefactor, además se utiliza un para enviar una letra la cual se utiliza para detener todo el proceso. Para enviar los datos tipo string (letras), es decir para escribir sobre la comunicación serial se utiliza a la salida de cada estructura de casos un VISA Write.

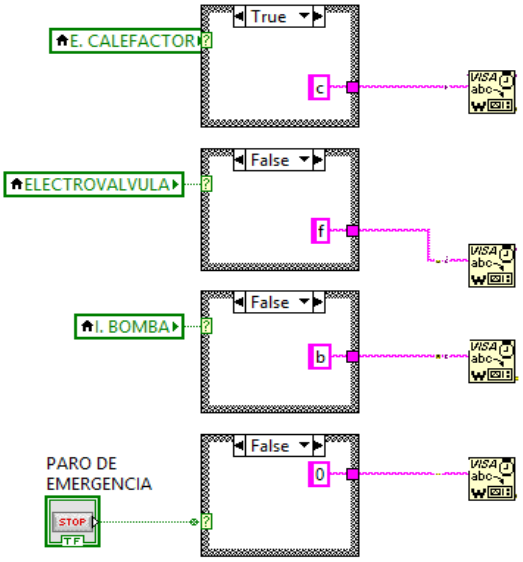


Figura 57 Escribir datos sobre la comunicación serial

Fuente: (Mallitasig, 2016)

Para finalizar se conecta la salida del VISA Configure Serial Port a las entradas de los VISA Read y los VISA Write luego las salidas de estos se conectan a la entrada de un VISA Close, para cerrar la comunicación serial y por último se conecta la salida del error a la entradas de cada VISA comenzando desde la configuración del puerto serial hasta el cierre de la comunicación. Todos los elementos que se incluyeron en el diagrama de bloques a excepción del VISA Configure Serial Port y de los VISA Close van dentro de una estructura While Loop que repite la programación cíclicamente.

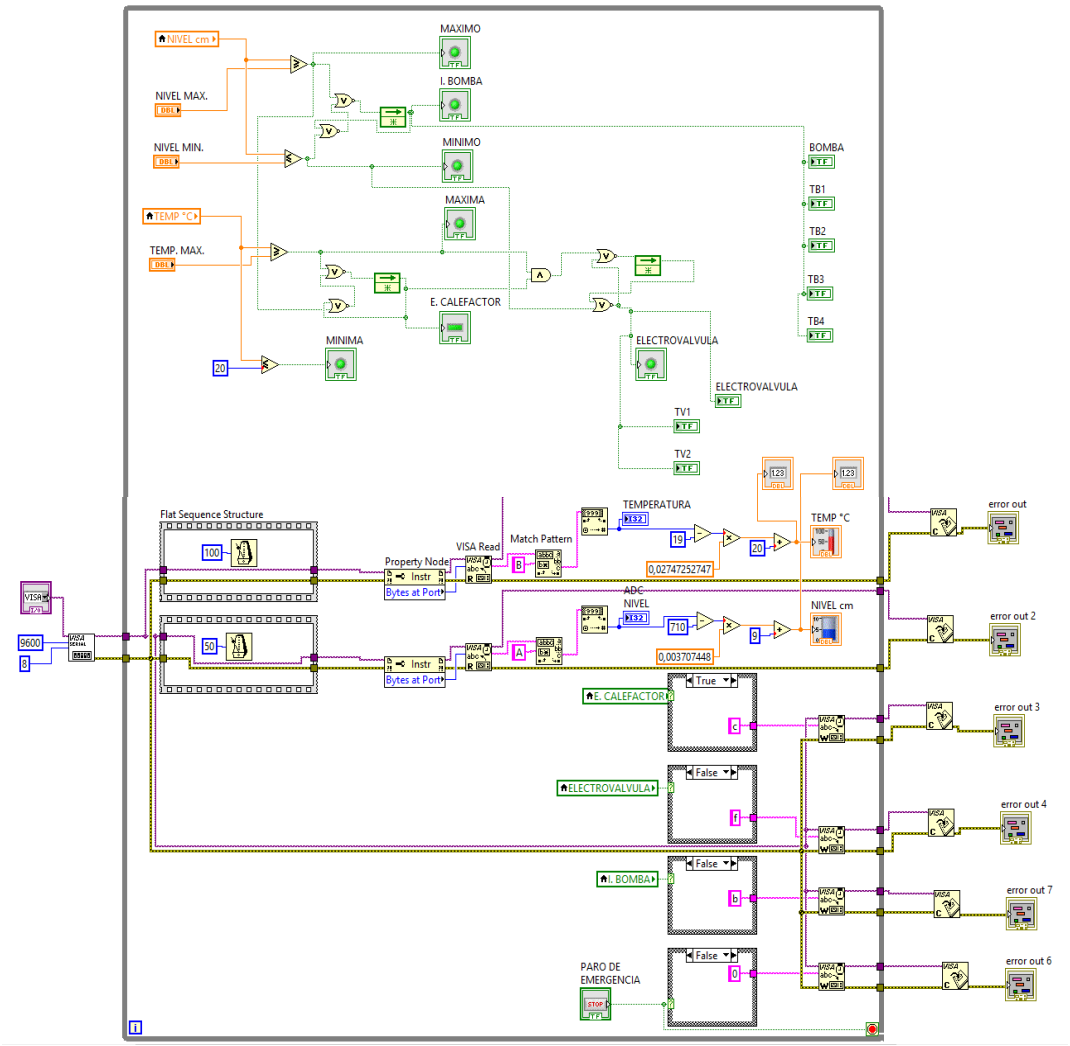


Figura 58 Programación en el diagrama de bloques de LabVIEW

Fuente: (Mallitasig, 2016)

3.16. Conexiones entre el Industruino I/O y el quipo PCT-3

Se procede a conectar en las dos salidas analogicas (CH1, CH2) y en la salida digital (CH1) del industruino los reles para manipular los elementos de control final, para esto los reles unicamente abren o cierran contactos para alimentar la boma, el elemento calefactor y la electrovalvula.

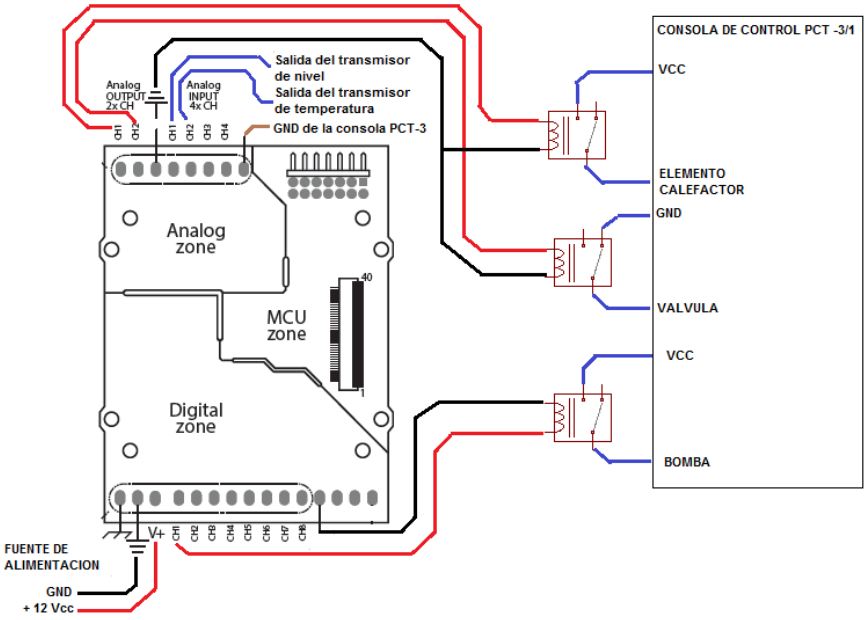


Figura 59 Conexiones entre el IND. I/O y la consola de control

Fuente: (Mallitasig, 2016)

CAPITULO IV

CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- Se recopiló información sobre el Industruino I/O en la página oficial del equipo, la cual permitió analizar las características técnicas así como la estructura interna y externa del equipo, además de conocer las prestaciones que brinda en un entorno industrial de esta manera se desarrolló el proceso automatizado de llenado y vaciado de un tanque.
- Se desarrolló el código de programación para adquirir las señales analógicas de corriente y voltaje de los transmisores de nivel y temperatura así como, las instrucciones para manipular los elementos de control final utilizando el lenguaje de programación y el entorno de desarrollo integrado de Arduino, además se utilizó la librería Indio y algunos bocetos de ejemplos que incluye la librería.
- Se estableció una comunicación serial entre la PC y el Industruino I/O, a una velocidad de transmisión/recepción de tres mil seiscientos baudios, mediante un cable micro USB para visualizar los valores de nivel y temperatura en la PC, también se enviaron de manera serial letras hacia el Industruino I/O para que ejecute las acciones necesarias para controlar el proceso.
- Se diseñó una interfaz hombre máquina utilizando el software LabVIEW para monitorear el proceso automatizado de llenado y vaciado del tanque, lo que permitió apreciar en tiempo real los valores del nivel y temperatura en los indicadores numéricos. Además, a través del panel frontal se puede fijar los valores máximos y mínimos así como parar el proceso en cualquier momento.

4.2. Recomendaciones

- Revisar la arquitectura interna así como el diagrama de pines que se encuentran en las hojas de datos del Industruino I/O, para conocer más sobre el funcionamiento y requerimientos técnicos del equipo.
- Realizar la calibración analógica del Industruino I/O para aumentar la precisión de las entradas y salidas analógicas usando los bosquejos que se encuentra disponible en el blog de la página de Industruino y seguir el procedimiento que se indica en el mismo. De no hacer ésta calibración no se podrá adquirir las señales de los transmisores conectados al Industruino.
- No desconectar el Industruino del puerto COM que se utilizó para cargar el programa desarrollado en el IDE de Arduino ya que éste establece la comunicación serial con el software de LabVIEW.
- Tener en cuenta las advertencias y las indicaciones presentes en la HMI con la finalidad de que el proceso se lleve a cabo de acuerdo a las instrucciones que ejecuta el Industruino.
- En la programación de LabVIEW utilizar diferentes tiempos de lectura para visualizar los valores de nivel y temperatura provenientes desde el controlador.

GLOSARIO DE TERMINOS

A

Arduino: Plataforma de software libre y fácil de programar, basada en una tarjeta con un microcontrolador.

ADC: Convertidor analógico digital.

B

Biestable: Almacena un estado lógico de acuerdo al estado de sus entradas.

C

Compilar: Buscar errores en las líneas de código de la programación, antes de ser grabada en la memoria del microcontrolador.

Comunicación serial: Medio para conectar dos dispositivos para intercambiar datos.

D

DAC: Convertidor digital analógico.

E

Entrada/salida analógica: Permiten leer o escribir variables analógicas de voltaje o corriente.

Entrada/salida digital: permiten leer o escribir estados lógicos en alto o bajo.

H

Hardware: Conjunto de partes físicas que constituyen un equipo de carácter tecnológico.

I

Interfaz: Medio para establecer una comunicación con un dispositivo.

ICSP: (In Circuit Serial Programming) pines que sirve para programar el BootLoader del Microcontrolador ATmega.

Implotar: Romperse con violencia un objeto hueco por exceso de presión exterior.

L

Lazo de control: Control y manejo de entadas y salidas en un proceso.

LabVIEW: (Laboratory Virtual Instrument Engineering Workbench), es un lenguaje de programación gráfica.

P

Proceso industrial: Conjunto de actividades desarrolladas para modificar las características de un objeto.

Processing: Lenguaje de programación basado en Java.

S

Sensor: Dispositivo que detecta magnitudes físicas y las convierte en variables eléctricas.

Software: Programas que ejecutan tareas en base a un sistema informático.

BIBLIOGRAFÍA

Admin. (29 de Septiembre de 2015). *ROBOLOGS.* Obtenido de Industruino, el Arduino para automatización industrial: http://robologs.net/2015/09/29/industruino-el-arduino-para-automatizacion-industrial/

Arduino. (2016). *Arduino.* Obtenido de Lenguaje de programación: https://www.arduino.cc/en/Reference/HomePage

Arduino. (2017). Obtenido de https://www.arduino.cc/en/Main/ArduinoBoardLeonardo

BANNER. (13 de Junio de 2016). Obtenido de U-GAGE S18U ULTRASONIC SERIES, ANALOG OUTPUT: http://info.bannerengineering.com/cs/groups/public/documents/literature/110738.pdf

Beltrán, C. (29 de Enero de 2011). *InfoPLC.* Obtenido de Colegio Salesiano de Concepción: http://www.infoplc.net/documentacion/12-instrumentacion-deteccion/54-medicion-de-temperatura

Blog Tecnosinergia. (14 de Agosto de 2014). *Blog Tecnosinergia.* Obtenido de ¿Qué es y para que sirve un relevador ?: https://tecnosinergiamx.com/2014/08/14/que-es-y-para-que-sirve-un-relevador/

Caroli, E. (S.f.). *Valvulas.* Obtenido de monografias.com: http://www.monografias.com/trabajos11/valvus/valvus.shtml

Control de procesos–Facet–Unt. (S.f.). *Elementos finales de control .* Obtenido de herrera.unt.edu.ar: http://www.herrera.unt.edu.ar/controldeprocesos/tema_3/tp3c.pdf

Creus, A. (2010). *Instrumentación industrial.* Barcelona,España: MARCOMBO S.A.

Digital I - R.M.M. (26 de Septiembre de 2016). *Introducción a los Controladores Lógicos Programables.* Obtenido de http://www.dsi.fceia.unr.edu.ar/downloads/digital_I/Apunte_PLC.pdf

Daniel Mogollón, C. R. (09 de Febrero de 2010). *Bomba.* Obtenido de unet.edu.ve: http://www.unet.edu.ve/~maqflu/doc/LAB-1-95.htm

Degem Systems. (S.f.).

Electricfor. (S.f.). *Resistencias para inmersión.* Obtenido de electricfor.es: http://www.electricfor.es/es/16523/Resistencias-para-inmersion.htm

Garcia, J. (2016). *Academia.edu.* Obtenido de http://www.academia.edu/7885227/Sistemas_de_control_-_lazo_abierto_-lazo_cerrado

GitHub, Inc. (2016). *Librerías Industruino.* Obtenido de https://github.com/Industruino/libraries#indio

Industruino. (2016). *Industruino*. Obtenido de https://industruino.com/

Jeison Torres, D. D. (8 de Septiembre de 2010). *es.slideshare.net.* Obtenido de http://es.slideshare.net/josueacerov/diapositivas-de-sensor-de-temperatura-jeison-torres-diego-diaz-jhonatan-mio

Kamp, W. V. (2008). *Teoría y práctica de medición de niveles.*

Mallitasig, A. (2016).

Mecánica de fluidos. (26 de Enero de 2008). Obtenido de Cavitación: http://ingenieros2011unefa.blogspot.com/2008/01/cavitacion.html

Medición de nivel de líquidos. (26 de Abril de 2011). Obtenido de webdelprofesor.ula.ve: http://webdelprofesor.ula.ve/ingenieria/oscaror/CursosDictados/web%20instrumentacion%20industrial/1%20transductores%20para%20procesos%20industriales/libro%20pdf/CAP%203%20Medicion_nivel_2009_n.pdf

Micro Capacitación. (18 de Julio de 2011). *WWW.MICRO.COM.AR.* Obtenido de Curso 061, Controlador Lógico Programable (PLC):

http://www.microautomacion.com/capacitacion/Manual061ControladorLgicoProgramablePLC.pdf

Micro JPM S.A. (2017). Obtenido de SRD-12VDC-SL-C Relay 12VDC SPDT: http://www.microjpm.com/products/srd-12vdc-sl-c-spdt-12vdc/

National Instruments. (2016). *LabVIEW*. Obtenido de ni.com: http://www.ni.com/labview/esa/

National, I. (2014). LabVIEW.

Navas, E. (S.f.). *Scribd.com.* Obtenido de Sensor: https://es.scribd.com/doc/50052577/Definicion-de-sensor

Omega Engineering. (2016). *es.omega.com*. Obtenido de Sensor de nivel: http://es.omega.com/prodinfo/sondas-de-nivel-medicion.html

Omega Engineering. (2017). *Termopozos.* Obtenido de es.omega.com: http://es.omega.com/technical-learning/criterios-de-seleccion-y-caracteristicas-de-termopozos.html

Orozco, S. (11 de Mayo de 2001). *Golpe de Ariete.* Obtenido de fluidos.eia.edu.co: http://fluidos.eia.edu.co/hidraulica/articuloses/flujoentuberias/golpedeariete/golpedeariete.html

Páez, O. (S.f.). Obtenido de Norma ISA: http://www.automaticausach.cl/asignaturas/controlautind/304_Norma_ISA_PID.pdf

PE, I. (29 de Julio de 2014). *ComoHacer.eu.* Obtenido de Análisis comparativo de las placas Arduino (oficiales y compatibles): http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/

Ramirez, C. (28 de Septiembre de 2005). *mailxmail.com.* Obtenido de Estructura básica de un PLC: http://www.mailxmail.com/curso-controladores-logicos-programables/estructura-basica-plc

Salvetti, D. (23 de Mayo de 2012). *Electrónica Industrial.* Obtenido de eet602ei.blogspot.com: http://eet602ei.blogspot.com/2012/05/sistemas-de-control-lazo-abiertocerrado.html

Sistemas de control. (20 de Febrero de 2016). Obtenido de upcommons.upc.edu: https://upcommons.upc.edu/bitstream/handle/2099.1/3330/34059-5.pdf?sequence=5

Tamayo, A. (21 de Junio de 2009). *Comunicación serial*. Obtenido de galaxi0.wordpress.com: https://galaxi0.wordpress.com/el-puerto-serial/

Tecnomedición. (2010). *Efecto Seebeck y Peltier.* Obtenido de tecnomedicion.com: http://www.tecnomedicion.com/inicio/efecto-seebeck-y-peltier.html

Turmero, P. (S.f.). *Automatización industrial*. Obtenido de Monografias.com: http://www.monografias.com/trabajos105/automatizacion-industrial-interfaz-hombre-maquina/automatizacion-industrial-interfaz-hombre-maquina.shtml

Tutorial LabVIEW. (22 de Septiembre de 2009). Obtenido de esi2.us.es: http://www.esi2.us.es/~asun/LCPC06/TutorialLabview.pdf

Uralita Sistemas de tubería. (S.f.). *La cavitación en sistemas de tuberías.* Obtenido de agronoms.cat: http://www.agronoms.cat/media/upload/editora_24/Cavitacion%20espa%C3%B1ol%202_editora_241_90.pdf

Uriza, M. (2016). *Diagramas de tuberia e instrumentación.* Obtenido de academia.edu: http://www.academia.edu/9376738/DIAGRAMAS_DE_TUBERIA_E_INSTRUMENTACION_DTI

Válvulas de solenoide. (30 de Diciembre de 2016). Obtenido de file:///C:/Users/Usuario/Downloads/funcionan%20valvulas%20s4bbd16c3832c9.pdf

Weebly. (S.f.). *arduinodhtics.weebly.com.* Obtenido de Arduino: Tecnología para todos: http://arduinodhtics.weebly.com/iquestqueacute-es.html

ANEXOS

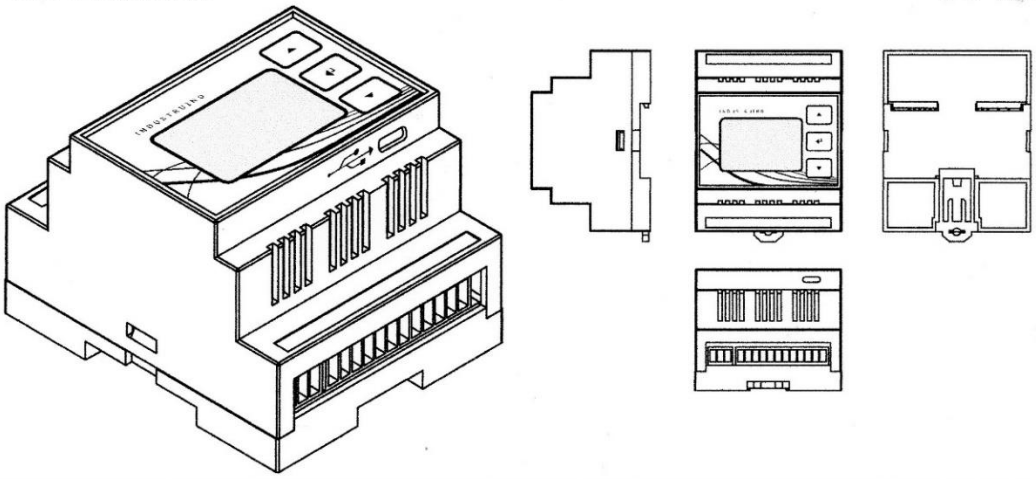
ÍNDICE DE ANEXOS

Anexo A

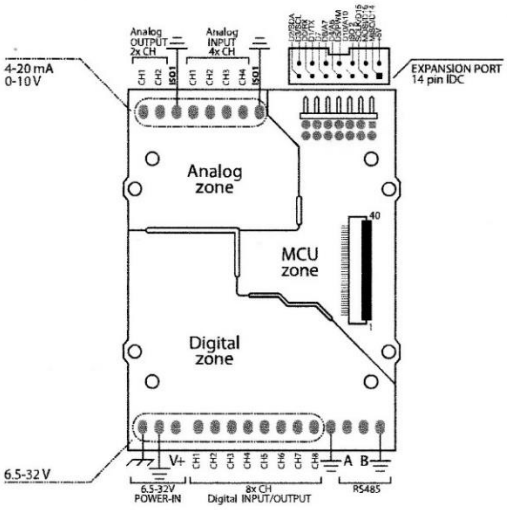
Datasheet del Industruino

Ind.I/O Datasheet

CE FC



Ind.I/O Baseboard Pinout



Notes

Industruino Ind.I/O is the Arduino-compatible equivalent of a PLC. The interface board offers 8 channels of 24V I/O, 4 channels of 0-10V/4-20mA 18bit ADC, 2 channels of 0-10V/4-20mA 12bit DAC, isolated RS485 transceiver, isolated power zones.

*Power for the main Industruino functions is supplied through the digital field section of the system. The MCU control section and analog field section are both galvanically isolated from the digital field section and each other. When a single power supply is used to power both Industruino and analog peripherals, the GND line of the analog section should be tied to the GND line of the digital section. If the GND line of the analog field section is not tied to the digital GND line, any incoming analog signals will appear to be floating.

*When operating in an electrically noisy environment it is possible to use a seperate power supply for your analog peripherals. This will improve the stability of your analog input/output signals. In this case the GND line of the digital field power supply should not be tied to the analog supply's GND line.

*The majority of IND.I/O board functions are controlled via I2c, therefore it requires the "Indio" Arduino library which can be found on our website's support page.

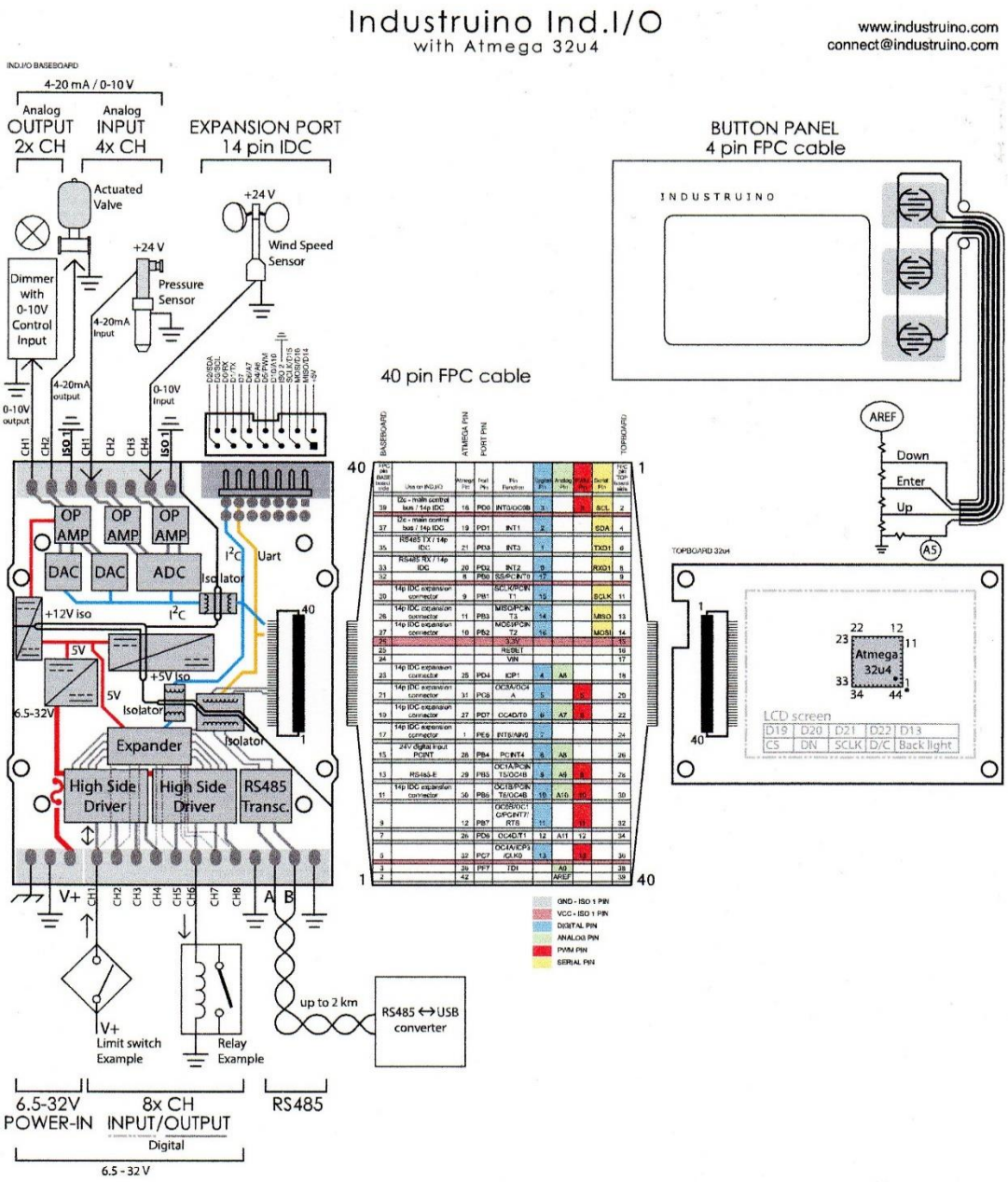
https://industruino.com/page/techcentre

Specifications subject to change without notice.
Date: 8.03.2015

Installation	
Mounting	on 35 mm DIN rail, 4 spacing units wide
Supply voltage	
Standard input voltage	12V / 24V
permissible range, lower limit (DC)	6.5 V
permissible range, upper limit (DC)	32 V
Digital inputs	
Number of digital inputs	8 (shared with digital outputs)
Type of digital input	Galvanically isolated serializer with interrupt
Input voltage range	0-32V
Logic HIGH voltage	>11V
Logic LOW voltage	<3V
Maximum trigger frequency	10 KHz
Protection of digital outputs	Short-circuit, over-current, over-temperature, ESD, transients.
Digital outputs	
Number of digital outputs	8 (shared with digital outputs)
Type of digital output	Galvanically isolated high-side driver (Charge pump NFET)
Output voltage range	Tied to supply voltage (6.5-32V)
Maximum current per output	2.5 A
Maximum total current	6.5 A
Maximum switching frequency	400 Hz
Protection of digital outputs	Short-circuit, over-current, over-temperature, ESD, transients.
Analog inputs	
Number of analog inputs	4
Type of analog inputs	Buffered ADC
Range of voltage measurement	0-10V
Range of current measurement	0-20mA
Switching of voltage / current mode	Automatic - in software
Resolution	18Bit
Conversion rate	18bit: 3.75 Hz - 16bit: 15 Hz - 14bit: 60 Hz - 12bit: 240 Hz
Protection of analog inputs	ESD, transients.
Analog Outputs	
Number of analog outputs	2
Type of analog outputs	Buffered DAC
Range of output voltage	0-10V
Range of output current	0-20mA
Switching of voltage / current mode	Automatic - in software
Resolution	12Bit
Update rate	20 KHz
Protection of analog outputs	Short-circuit, over-current, over-temperature, ESD, transients.
Communication ports	
RS485	
Isolation topology	Isolated from MCU and analog field section
Duplex type	Half duplex
Number of receivers on bus	32
Data rate	1 Mbps
Expansion port (direct MCU control)	
Isolation topology	Isolated from digital and analog field section
Number of pins	14
Voltage level	5V
Protocols supported	SPI, I2C, UART, 9 GPIO's
Protection of expansion port	ESD, transients.
User interface	
LCD	128x64 pixel FSTN with dimmable backlight
Push buttons	3 - push button membrane panel
Environment	
Protection class	IP20
Ambient operating temperature	0 - 55 °C
Dimensions	
Width	71.5 mm
Height	87 mm
Depth	58 mm
Weight	150 g

Anexo B

Diagrama del Industruino



IND.I/O is an Arduino IDE programmable controller.

*"Power-in /digital field zone", "Analog field zone" and "MCU zone" are isolated from each other. Where single power supply is used for both power-in and analog peripherals the respective GND's need to be tied together.

*The majority of IND.I/O board functions are controlled via i2c, therefore it requires the "Indio" library which can be found on our website.

Ind.I/O Baseboard
8 CH 24V digital I/O
4 CH 0-10V/4-20mA analog IN
2 CH 0-10V/4-20mA analog OUT
Isolated RS485 transceiver
3 isolated supply rails.

Topboard 32u4
Atmega 32u4 MCU
Flash 32 KB
SRAM 2.5 KB
EEPROM 1 KB.

INDUSTRUINO

Arduino libraries are available for download on our website: www.industruino.com

Industruino is an Arduino compatible industruial controller with LCD screen, housed in a robust DIN-rail casing.

Anexo C

Programación para controlar el proceso

```
//---LIBRERIAS---

#include <Wire.h>

#include <Indio.h>

//---VARIABLES---

 char dato;

//---INICIALIZACION  DEL PROGRAMA---

void setup() {

// put your setup code here, to run once:

//---COMUNICACION_SERIAL---

Serial.begin (9600); // Abre el puerto TX / RX COMUNICACION  SERIE

//---MODO  DE SALIDAS ANALOGICAS---

//---ACTIVAR_RELE_ELECTROVALVULA---

Indio.analogWriteMode(1, V10); // FIJA CH1 COMO SALIDA ANALOGICA
10V

Indio.analogWrite(1,0,false);  //INICIALIZA  CH1 A 0V

//---ACTIVAR_RELE_NIQUELINA--

Indio.analogWriteMode(2, V10); // FIJA CH1 COMO SALIDA ANALOGICA
10V

Indio.analogWrite(2,0,false);  //INICIALIZA  CH2 A 0V

//--ACTIVAR_RELE_BOMBA

Indio.digitalMode(1,OUTPUT);

Indio.digitalWrite(1,LOW);


//---RESOLUCION_DEL_ADC---
```

```
Indio.setADCResolution(12);  //RESOLUCION  DEL ADC
//---MODO DE LECTURA DE LA ENTRADA----
// NIVEL
Indio.analogReadMode(1,mA_raw); //FIJA EL CH1 A ENTRADA ANALOGICA mA_raw
//TEMP
Indio.analogReadMode(2,V10_raw);  //FIJA  EL  CH2  A  ENTRADA ANALOGICA 10V_raw
}
//----PROGRAMA  PRINCIPAL-----
void loop() {
 // put your main code here, to run repeatedly:
 Indio.digitalMode(1,LOW);
//-----NIVEL-TEMP--
String CH1=String(Indio.analogRead(1),DEC);  //LEE NIVEL
String CH2=String(Indio.analogRead(2),DEC);  //LEE TEMP
String datos1=String("A"+CH1); //NIVEL
String datos2=String("B"+CH2); //TEMP
Serial.print(datos1);// DATOS NIVEL
Serial.print(datos2);// DATOS TEMP
//---ON-OFF-BOMBA-ELECTROVALVULA-NIQUELINA----
if(Serial.available()>0)   // pregunta si existe un valor diferente de cero
{
 dato=Serial.read();  //lee el dato
 //---BOMBA----
```

```
if (dato=='a')
{
Indio.digitalWrite(1,HIGH);
}
if (dato=='b')
{
Indio.digitalWrite(1,LOW);
}
//---NIQUELINA---
 if (dato=='c')
{
Indio.analogWrite(1,  10,false);
}
if (dato=='d')
{
Indio.analogWrite(1,  0,false);
}
//---ELECTROVALVULA---
if (dato=='e')
{
Indio.analogWrite(2,  10,false);
}
if (dato=='f')
{
```

```
Indio.analogWrite(2,  0,false);
}
if (dato=='p')
{
paro();
}
}
}
//--SUBRUTINA—PARO---
void paro() {
Indio.digitalWrite(1,LOW);
Indio.analogWrite(2,  0,false);
Indio.analogWrite(1,  0,false);
if(Serial.available()>0)    // Pregunta si existe un valor diferente de cero
{
dato=Serial.read();  //lee el dato
if (dato=='p') {
paro();
}
}
}
```

Anexo D

Programación para la calibración analógica

```
/*
      * Industruino Demo Code - IND.I/O analog I/O calibration
      * universal version for 1286 and 32u4 (does not use membrane buttons)
      *
      * INSTRUCTIONS:
      *
      * Connect your INDIO to an external power supply and to your laptop via
      USB
      * Upload this sketch, open the Serial Monitor, and follow the
      instructions
      *
      * Calibration data will appear in the Serial Monitor after each step
      (ADC_voltage, ADC_current, DAC_voltage, DAC_current)
      *
      * The resulting calibration data have to replace the default values in
      your Indio library file Indio.cpp (approximately lines 41 to 75)
      * There are 4x 4 lines and each block of 4 lines refers to one type of
      I/O: ADC_voltage, ADC_current, DAC_voltage, DAC_current
      * After each step the Serial Monitor shows the calibration data for that
      step; use these to update the corresponding 4 lines in Indio.cpp
      *
      * The USB connection may drop at the end of step 4, this is a known issue
      but does not affect the functionality of this sketch
      * Tom Tobback for Industruino - September 2016
      *
      *
      * Permission is hereby granted, free of charge, to any person obtaining a
      copy
      * of this software and associated documentation files (the "Software"),
      to deal
      * in the Software without restriction, including without limitation the
      rights
      * to use, copy, modify, merge, publish, distribute, sublicense, and/or
      sell
      * copies of the Software, and to permit persons to whom the Software is
      * furnished to do so, subject to the following conditions:
      *
      * The above copyright notice and this permission notice shall be included
      in
      * all copies or substantial portions of the Software.
      *
```

```
* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS
OR
* IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY,
* FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
THE
* AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
* LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM,
* OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS
IN
* THE SOFTWARE.
*/


#include <Indio.h>
#include <Wire.h>


#include <UC1701.h>
static UC1701 lcd;


// These constants won't change
const int backlightIntensity = 0;        // LCD backlight intensity
0=full on


const int inputChannels = 4;      // number of input channels
const int outputChannels = 2;     // number of output channels


int ADC_current_low_raw[inputChannels];     // arrays to hold calibration
data per channel for Low and High voltage and current
int ADC_current_low_uA[inputChannels];
int ADC_current_high_raw[inputChannels];
int ADC_current_high_uA[inputChannels];


int ADC_voltage_low_raw[inputChannels];
int ADC_voltage_low_mV[inputChannels];
int ADC_voltage_high_raw[inputChannels];
int ADC_voltage_high_mV[inputChannels];


int DAC_current_low_raw[outputChannels];
```

```
int DAC_current_low_uA[outputChannels];
int DAC_current_high_raw[outputChannels];
int DAC_current_high_uA[outputChannels];


int DAC_voltage_low_raw[outputChannels];
int DAC_voltage_low_mV[outputChannels];
int DAC_voltage_high_raw[outputChannels];
int DAC_voltage_high_mV[outputChannels];


byte i = 0;                   // counter


////////////////////////////////////////////////////////////////////////////
//////////////////////////////////


void setup() {


  pinMode(13, OUTPUT); //set backlight pin to output   for 32u4
  analogWrite(13, backlightIntensity);
  pinMode(26, OUTPUT); //set backlight pin to output   for 1286
  analogWrite(26, backlightIntensity);


  lcd.begin();
  lcd.clear();


  screenWelcome(); //load first menu


  Serial.begin(9600);
  while (!Serial) {};


  serialMenu();
}


////////////////////////////////////////////////////////////////////////////
//////////////////////////////////
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////
```

```
void loop() {


  int option = Serial.parseInt();
  if (option < 1 || option > 4) return;
  else {
    Serial.println(option);
    switch (option) {
      case 1:
        DAC_voltage();
        break;
      case 2:
        DAC_current();
        break;
      case 3:
        ADC_voltage();
        break;
      case 4:
        ADC_current();
        break;
    }
  }
}




////////////////////////////////////////////////////////////////////////
//////////////////////////////////
////////////////////////////////////////////////////////////////////////
//////////////////////////////////


void ADC_voltage() {


  Serial.println(F("ADC calibration voltage mode: 0-10V"));
  Serial.println(F("================================="));
  screenADC_voltage_low();


  Indio.setADCResolution(16);
  for (i = 1; i <= inputChannels; i++) {
    Indio.analogReadMode(i, V10_raw);
```

```
  }


  Serial.println();
  Serial.println(F("Connect known LOW voltage to all analog input
channels, e.g. 2500mV"));
  Serial.print(F("Enter the voltage (in mV) in the Serial Monitor box
above. Received (mV): "));
  int voltage = getSerialInt(0, 10000);
  Serial.println(voltage);


  for (i = 1; i <= inputChannels; i++) {
    ADC_voltage_low_mV[i]  = voltage;
    ADC_voltage_low_raw[i] = Indio.analogRead(i);
// read Analog IN at same time
  }


  Serial.println();
  Serial.print(F("Calibration data ADC_voltage_low_mV: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_voltage_low_mV[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_voltage_low_mV[inputChannels]);


  Serial.print(F("Calibration data ADC_voltage_low_raw: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_voltage_low_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_voltage_low_raw[inputChannels]);


  screenADC_voltage_high();
  Serial.println();
  Serial.println(F("Connect known HIGH voltage to all analog input
channels, e.g. 7500mV"));
  Serial.print(F("Enter the voltage (in mV) in the Serial Monitor box
above. Received (mV): "));
  voltage = getSerialInt(0, 10000);
  Serial.println(voltage);
```

```
  for (i = 1; i <= inputChannels; i++) {
    ADC_voltage_high_mV[i]  = voltage;
    ADC_voltage_high_raw[i] = Indio.analogRead(i);
// read Analog IN at same time
  }


  Serial.println();
  Serial.print(F("Calibration data ADC_voltage_high_mV: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_voltage_high_mV[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_voltage_high_mV[inputChannels]);


  Serial.print(F("Calibration data ADC_voltage_high_raw: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_voltage_high_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_voltage_high_raw[inputChannels]);


  screenDone();
  serialMenu();
}


////////////////////////////////////////////////////////////////////////////
////////////////////////////////


void ADC_current() {
  Serial.println(F("ADC calibration current mode: 4-20mA"));
  Serial.println(F("==================================="));
  Serial.println();


  Indio.setADCResolution(12);
  for (i = 1; i <= inputChannels; i++) {
    Indio.analogReadMode(i, mA_raw);
  }


  screenADC_current();
```

```
  for (i = 1; i <= inputChannels; i++) {
// read Analog IN one by one


    Serial.print(F("Connect a known LOW current to input channel "));
    Serial.print(i);
    Serial.println(F(", e.g. 10000uA"));
    Serial.print(F("Enter the voltage (in uA) in the Serial Monitor box
above. Received (uA): "));
    ADC_current_low_uA[i] = getSerialInt(0, 20000);
    Serial.print(ADC_current_low_uA[i]);


    ADC_current_low_raw[i] = Indio.analogRead(i);
    Serial.print(F(" Measured (raw): "));
    Serial.println(ADC_current_low_raw[i]);
    //    Serial.println();
  }


  Serial.println();
  Serial.print(F("Calibration data ADC_current_low_uA: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_current_low_uA[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_current_low_uA[inputChannels]);


  Serial.print(F("Calibration data ADC_current_low_raw: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_current_low_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_current_low_raw[inputChannels]);
  Serial.println();




  for (i = 1; i <= inputChannels; i++) {
// read Analog IN one by one
```

```
    Serial.print(F("Connect a known HIGH current to input channel "));
    Serial.print(i);
    Serial.println(F(", e.g. 20000uA"));
    Serial.print(F("Enter the voltage (in uA) in the Serial Monitor box
above. Received (uA): "));
    ADC_current_high_uA[i] = getSerialInt(0, 20001);
    Serial.print(ADC_current_high_uA[i]);


    ADC_current_high_raw[i] = Indio.analogRead(i);
    Serial.print(F(" Measured (raw): "));
    Serial.println(ADC_current_high_raw[i]);
    //    Serial.println();
  }




  Serial.println();
  Serial.print(F("Calibration data ADC_current_high_uA: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_current_high_uA[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_current_high_uA[inputChannels]);


  Serial.print(F("Calibration data ADC_current_high_raw: "));
  for (i = 1; i < inputChannels; i++) {
    Serial.print(ADC_current_high_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(ADC_current_high_raw[inputChannels]);


  ////////////////// WILL HANG HERE  ////////////////////////
  screenDone();      //// NEXT LCD DISPLAY HANGS
  serialMenu();


}
```

```
////////////////////////////////////////////////////////////////////////
////////////////////////////////

void DAC_voltage() {
  Serial.println(F("DAC calibration voltage mode: 0-10V"));
  Serial.println(F("================================="));
  Serial.println();


  for (i = 1; i <= outputChannels; i++) {
    Indio.analogWriteMode(i, V10_raw);
    DAC_voltage_low_raw[i] = 1000;
    Indio.analogWrite(i, DAC_voltage_low_raw[i], true);
  }


  for (i = 1; i <= outputChannels; i++) {
    screenDAC_voltage();
    Serial.print(F("Measure the VOLTAGE (in mV) on channel "));
    Serial.println(i);
    Serial.print(F("Enter the voltage (in mV) in the Serial Monitor box
above. Received (mV): "));
    DAC_voltage_low_mV[i] = getSerialInt(0, 10000);
    Serial.println(DAC_voltage_low_mV[i]);
  }


  Serial.println();
  Serial.print(F("Calibration data DAC_voltage_low_raw: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_voltage_low_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_voltage_low_raw[outputChannels]);


  Serial.print(F("Calibration data DAC_voltage_low_mV: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_voltage_low_mV[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_voltage_low_mV[outputChannels]);
  Serial.println();
```

```
  for (i = 1; i <= outputChannels; i++) {
    DAC_voltage_high_raw[i] = 3600;
    Indio.analogWrite(i, DAC_voltage_high_raw[i], true);
  }


  for (i = 1; i <= outputChannels; i++) {
    screenDAC_voltage();
    Serial.print(F("Measure the VOLTAGE (in mV) on channel "));
    Serial.println(i);
    Serial.print(F("Enter the voltage (in mV) in the Serial Monitor box
above. Received (mV): "));
    DAC_voltage_high_mV[i] = getSerialInt(0, 15000);
    Serial.println(DAC_voltage_high_mV[i]);
  }


  Serial.println();
  Serial.print(F("Calibration data DAC_voltage_high_raw: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_voltage_high_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_voltage_high_raw[outputChannels]);


  Serial.print(F("Calibration data DAC_voltage_high_mV: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_voltage_high_mV[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_voltage_high_mV[outputChannels]);
  Serial.println();


  screenDone();
  serialMenu();


}


////////////////////////////////////////////////////////////////////////
//////////////////////////////////
```

```
void DAC_current() {
  Serial.println(F("DAC calibration current mode: 4-20mA"));
  Serial.println(F("==================================="));


  for (i = 1; i <= outputChannels; i++) {
    Indio.analogWriteMode(i, mA_raw);
    DAC_current_low_raw[i] = 1000;
    Indio.analogWrite(i, DAC_current_low_raw[i], true);
  }


  for (i = 1; i <= outputChannels; i++) {
    screenDAC_current();
    Serial.print(F("Measure the current (in uA) on channel "));
    Serial.println(i);
    Serial.print(F("Enter the current (in uA) in the Serial Monitor box
above. Received (uA): "));
    DAC_current_low_uA[i] = getSerialInt(0, 20000);
    Serial.println(DAC_current_low_uA[i]);
  }


  Serial.println();
  Serial.print(F("Calibration data DAC_current_low_raw: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_current_low_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_current_low_raw[outputChannels]);


  Serial.print(F("Calibration data DAC_current_low_uA: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_current_low_uA[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_current_low_uA[outputChannels]);
  Serial.println();


  for (i = 1; i <= outputChannels; i++) {
    DAC_current_high_raw[i] = 3600;
    Indio.analogWrite(i, DAC_current_high_raw[i], true);
  }
```

```
  for (i = 1; i <= outputChannels; i++) {
    screenDAC_current();
    Serial.print(F("Measure the CURRENT (in uA) on channel "));
    Serial.println(i);
    Serial.print(F("Enter the current (in uA) in the Serial Monitor box
above. Received (uA): "));
    DAC_current_high_uA[i] = getSerialInt(0, 30000);
    Serial.println(DAC_current_high_uA[i]);
  }


  Serial.println();
  Serial.print(F("Calibration data DAC_current_high_raw: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_current_high_raw[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_current_high_raw[outputChannels]);


  Serial.print(F("Calibration data DAC_current_high_uA: "));
  for (i = 1; i < outputChannels; i++) {
    Serial.print(DAC_current_high_uA[i]);
    Serial.print(F(","));
  }
  Serial.println(DAC_current_high_uA[outputChannels]);
  Serial.println();


  screenDone();
  serialMenu();
}


////////////////////////////////////////////////////////////////////////////
////////////////////////////////


void screenWelcome() { //this function draws the first menu - splash
screen


  lcd.clear();
  lcd.setCursor(15, 1); //set the cursor to the fifth pixel from the left
edge, third row.
```

```
  lcd.print(F("Welcome to")); //print text on screen
  lcd.setCursor(15, 2); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("Industruino!")); //print text on screen
  lcd.setCursor(15, 4); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("Analog I/O")); //print text on screen
  lcd.setCursor(15, 5); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("calibration:")); //print text on screen
  lcd.setCursor(15, 6); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("see Serial Monitor")); //print text on screen
  delay(2000);
}


////////////////////////////////////////////////////////////////////////
//////////////////////////////


void screenDone() {


  lcd.clear();
  lcd.setCursor(15, 1); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("To continue")); //print text on screen
  lcd.setCursor(15, 2); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("Analog I/O")); //print text on screen
  lcd.setCursor(15, 3); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("calibration:")); //print text on screen
  lcd.setCursor(15, 5); //set the cursor to the fifth pixel from the left
edge, third row.
  lcd.print(F("see Serial Monitor")); //print text on screen
  delay(2000);


}


////////////////////////////////////////////////////////////////////////
//////////////////////////////
```

```
void screenADC_voltage_low() {


  lcd.clear();
  lcd.setCursor(5, 1);
  lcd.print(F("ANALOG IN 0-10V"));
  lcd.setCursor(5, 3);
  lcd.print(F("connect LOW voltage"));
  lcd.setCursor(5, 4);
  lcd.print(F("input Serial Monitor"));
}


////////////////////////////////////////////////////////////////////////
//////////////////////////////////


void screenADC_voltage_high() {


  lcd.clear();
  lcd.setCursor(5, 1);
  lcd.print(F("ANALOG IN 0-10V"));
  lcd.setCursor(5, 3);
  lcd.print(F("connect HIGH voltage"));
  lcd.setCursor(5, 4);
  lcd.print(F("input Serial Monitor"));
}


////////////////////////////////////////////////////////////////////////
//////////////////////////////////


void screenADC_current() {


  //  Serial.println("in screen");    for debugging
  //  Serial.println(freeRam());
  lcd.clear();
  lcd.setCursor(5, 1);
  lcd.print(F("ANALOG IN 4-20mA"));
  lcd.setCursor(5, 3);
  lcd.print(F("connect current"));
  lcd.setCursor(5, 4);
```

```
  lcd.print(F("to channels 1-4 "));
  //  lcd.print(i);
  //  lcd.setCursor(5, 6);
  //  lcd.print(F("input Serial Monitor"));
  //  Serial.println(freeRam());
}


////////////////////////////////////////////////////////////////////////
//////////////////////////////////


void screenDAC_voltage() {


  lcd.clear();
  lcd.setCursor(5, 1);
  lcd.print(F("ANALOG OUT 0-10V"));
  lcd.setCursor(5, 3);
  lcd.print(F("measure mV on"));
  lcd.setCursor(5, 4);
  lcd.print(F("channel "));
  lcd.print(i);
  lcd.setCursor(5, 6);
  lcd.print(F("input Serial Monitor"));
}


////////////////////////////////////////////////////////////////////////
//////////////////////////////////


void screenDAC_current() {


  lcd.clear();
  lcd.setCursor(5, 1);
  lcd.print(F("ANALOG OUT 4-20mA"));
  lcd.setCursor(5, 3);
  lcd.print(F("measure uA on"));
  lcd.setCursor(5, 4);
  lcd.print(F("channel "));
  lcd.print(i);
  lcd.setCursor(5, 6);
  lcd.print(F("input Serial Monitor"));
}
```

```
////////////////////////////////////////////////////////////////////////
////////////////////////////////

void serialMenu() {


  Serial.println();

Serial.println(F("==================================================
=========================="));
  Serial.println();
  Serial.println(F("Welcome to Industruino!"));
  Serial.println(F("Analog I/O calibration for IND.I/O"));
  Serial.println();
  Serial.println(F("Main menu"));
  Serial.println(F("1. DAC: analog OUT 0-10V"));
  Serial.println(F("2. DAC: analog OUT 4-20mA"));
  Serial.println(F("3. ADC: analog IN 0-10V"));
  Serial.println(F("4. ADC: analog IN 4-20mA"));


  Serial.println();
  Serial.println(F("Please enter your choice 1,2,3,4 in the Serial input
box above and hit return"));
  //  Serial.println(F("Make sure your Serial Monitor is configured for
'NO LINE ENDING'"));
  Serial.println();
}


////////////////////////////////////////////////////////////////////////
////////////////////////////////


int getSerialInt(int min, int max) {
  int number = -1;
  while (number <= min || number >= max) {
    while (Serial.available() == 0) { }
    number = Serial.parseInt();
  }
  return number;
}
```

```
////////////////////////////////////////////////////////////////////////////
//////////////////////////////////

int freeRam () {
  extern int __heap_start, *__brkval;
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int)
__brkval);
}
```

CURRICULUM VITAE

DATOS PERSONALES

Apellidos:	Mallitasig Mallitasig
Nombres:	Alex Xavier
Documento de identidad:	0503975062
Fecha de nacimiento:	23 de junio de 1993
Lugar de nacimiento:	Cotopaxi, La Matriz
Estado civil:	soltero
Dirección:	Av. Amazonas, Barrio Colaisa
Ciudad:	Latacunga
Teléfono:	032 385 842
Celular:	0979172020
E-mail:	alexmallitasig1993@gmail.com

ESTUDIOS REALIZADOS

Primaria

Escuela “Simón Bolívar”

Secundaria

Instituto Superior Tecnológico “Ramón Barba Naranjo”

Superior

Unidad de Gestión de Tecnologías de la Universidad de las Fuerzas Armadas – ESPE.

TÍTULOS OBTENIDOS

Bachiller Técnico Industrial en Electrónica de Consumo

EXPERIENCIA LABORAL

PRÁCTICAS PRE-PROFESIONALES

Corporación Nacional de Telecomunicaciones.

Función: Técnico de mantenimiento.

Fecha: 07/03/2016 al 08/04/2016.

Grupo de Aviación del Ejército N° 43 “Portoviejo”.

Función: Técnico de mantenimiento

Fecha: 25/08/2015 al 01/10/2015.

ACEPTACIÓN DEL USUARIO

Latacunga, Febrero del 2017

Yo, ING PABLO PILATASIG en calidad de encargado del Laboratorio de Instrumentación Virtual de la Unidad de Gestión de Tecnologías, me permito informar lo siguiente:

El proyecto de graduación elaborado por el Sr. **MALLITASIG MALLITASIG ALEX XAVIER,** con el tema: **"IMPLEMENTACIÓN DE UN INDUSTRUINO PARA PRÁCTICAS DE CONTROL DE PROCESOS EN EL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL DE LA UNIDAD DE GESTIÓN DE TECNOLOGÍAS",** ha sido efectuado de forma satisfactoria en las dependencias de mi cargo y que la misma cuenta con todas las garantías de funcionamiento, por lo cual extiendo este aval que respalda el trabajo realizado por el mencionado estudiante.

Por tanto me hago cargo de todas las instalaciones realizadas por el Sr. estudiante.

Atentamente,

ING. PABLO PILATASIG
ENCARGADO DEL LABORATORIO DE INSTRUMENTACIÓN VIRTUAL

HOJA DE LEGALIZACIÓN DE FIRMAS

DEL CONTENIDO DE LA PRESENTE INVESTIGACIÓN SE RESPONSABILIZA EL AUTOR

Mallitasig Mallitasig Alex Xavier

DIRECTOR DE LA CARRERA DE ELECTRÓNICA MENCIÓN INSTRUMENTACIÓN & AVIÓNICA

Ing. Pablo Pilatasig

Latacunga, Febrero del 2017