ESPE

UNIVERSIDAD DE LAS FUERZAS ARMADAS

INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA E INSTRUMENTACIÓN

TEMA: TECNOLOGÍAS DE BAJO COSTO PARA LA IMPLEMENTACIÓN DE CONTROL AVANZADO EN UN PROCESO INDUSTRIAL

AUTORES:

JORGE LUIS BUELE LEÓN

JOHN JAVIER ESPINOZA MEJÍA

DIRECTOR: ING. MARCO PILATÁSIG

LATACUNGA

2018

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Certifico que el trabajo de titulación, **"TECNOLOGÍAS DE BAJO COSTO PARA LA IMPLEMENTACIÓN DE CONTROL AVANZADO EN UN PROCESO INDUSTRIAL"** realizado por los señores **JORGE LUIS BUELE LEÓN Y JOHN JAVIER ESPINOZA MEJÍA**, 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 las Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar a los señores **JORGE LUIS BUELE LEÓN Y JOHN JAVIER ESPINOZA MEJÍA** para que lo sustenten públicamente.

Latacunga, 15 de enero de 2018

Ing. Marco Pilatásig
DIRECTOR

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

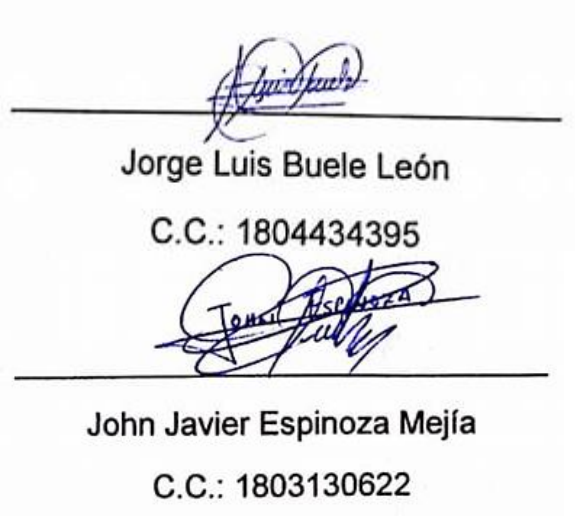
CARRERA DE INGENIERÍA EN ELECTRÓNICA E INSTRUMENTACIÓN

AUTORÍA DE RESPONSABILIDAD

Nosotros, **Jorge Luis Buele León**, con cédula de identidad N°1804434395 y **John Javier Espinoza Mejía**, con cédula de identidad N°1803130622 declaramos que el presente trabajo de titulación, **"TECNOLOGÍAS DE BAJO COSTO PARA LA IMPLEMENTACIÓN DE CONTROL AVANZADO EN UN PROCESO INDUSTRIAL"** 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 declaramos que este trabajo es de nuestra autoría, en virtud de ello nos declaramos responsables del contenido, veracidad y alcance de la investigación mencionada.

Latacunga, 15 de enero de 2018



DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA DE ELECTRÓNICA E INSTRUMENTACIÓN

AUTORIZACIÓN

Nosotros, **JORGE LUIS BUELE LEÓN y JOHN JAVIER ESPINOZA MEJÍA**, autorizamos a la Universidad de las Fuerzas Armadas ESPE publicar en la Biblioteca Virtual de la institución el presente trabajo de titulación **"TECNOLOGÍAS DE BAJO COSTO PARA LA IMPLEMENTACIÓN DE CONTROL AVANZADO EN UN PROCESO INDUSTRIAL"** cuyo contenido, ideas y criterios son de nuestra autoría y responsabilidad.

Latacunga, 15 de enero de 2018

Jorge Luis Buele León

C.C.: 1804434395

John Javier Espinoza Mejía

C.C.: 1803130622

DEDICATORIA

La elaboración de este trabajo, un texto que simboliza años de estudio y vivencias, es dedicado a Dios, quien me colocó en el seno de un maravilloso hogar. A mis padres, hermano y demás familiares por tanto amor y apoyo que recibo cada día y a mis amigos y personas que aprecio, quienes han puesto su granito de arena a lo largo de este proceso.

Jorge Luis Buele

DEDICATORIA

Este trabajo va dedicado en primer lugar a Dios, quien a lo largo de mi camino como estudiante a guiado mis pasos de forma correcta, luego quiero dedicárselo a mis padres “Enma y Rufo” y mis hermanas “Karen y Carla”, personas que han sido los pilares fundamentales para que nunca me dé por vencido en este camino tan complicado del estudiante; este trabajo es la culminación de una meta en mi vida, la cual no hubiera sido alcanzada sin el apoyo de toda mi familia; a ellos quiero decirles Gracias y que este logro es para ustedes.

John Espinoza

AGRADECIMIENTO

Extiendo mi total y profundo agradecimiento a Dios Todopoderoso, por haberme brindado la oportunidad de educarme y darme salud y vida cada día.

A mis padres Luis y Ana, por el apoyo, amor y paciencia que han tenido hacia mí desde que tengo memoria e incluso antes de ello, a pesar de los errores que he sabido cometer.

A mi hermano Pablo, por el cariño que me brinda y porque ha estado junto a mi cuando lo he requerido, ayudándome sin objeción.

A todos mis familiares, que, a pesar de no estar junto a mí, a diario están pendientes y se hacen presentes con sus consejos.

A los docentes universitarios que fueron parte de mi formación profesional. De manera especial al Ing. Marco Pilatásig por su predisposición a ser parte de este proyecto y motivarnos a realizar otras actividades a la par. Además, a los ingenieros Franklin Silva y Edwin Pruna, quienes han sido partícipes directos de este trabajo realizando las contribuciones pertinentes. Y muy encarecidamente a los ingenieros José Bucheli y Eddie Galarza quienes de igual manera han sabido transmitir sus conocimientos y recomendaciones.

Y finalmente con mucho énfasis, a todos mis amigos y personas que aprecio, quienes me han apoyado y han estado a mi lado cuando lo he necesitado, de quienes he recibido una palabra de motivación, un abrazo o un presente, les haré llegar mi agradecimiento personalmente uno a uno.

Jorge Luis Buele

AGRADECIMIENTO

Quiero agradecer a Dios, por ser mi guía en este camino que no llega a su fin pero que si significa el logro de una meta planteada desde el colegio. También quiero agradecer infinitamente a mis padres Enma y Rufo, ya que sin ellos no hubiera logrado cumplir este objetivo, ellos han sido las personas que constantemente han estado ahí para darme sus consejos y sobre todo apoyo para los momentos buenos y malos. A mis hermanas Karen y Carla, quiero agradecerles por ser parte de mi vida, ellas han sido el motivo para que me esfuerce, ya que soy su modelo a seguir. Quiero decirles que sin importar lo duro que sea cumplir sus objetivos, nunca se rindan, peleen hasta lograrlo. La peor derrota que puede existir en sus vidas es aquella en la cual no lo intentan por miedo. Gracias por brindarme ese amor de familia, ingrediente que considero fundamental en la vida de una persona para ser feliz.

Quiero agradecer a todos mis abuelitos, porque siempre estuvieron pendientes de mi alimentación u otras necesidades, en especial cuando vivía solo en Latacunga; en general quiero agradecer a todos mis familiares que de una u otra forma aportaron con su granito de arena para que pueda cumplir mi meta de convertirme en un profesional.

También quiero agradecer a mis amigos con los cuales he compartido esos momentos de diversión y sobre todo esos momentos de malas noches, gracias a la colaboración de cada uno de ellos todos pudimos salir adelante cada semestre en la Universidad; a alguno de ellos conocí desde el inicio de la carrera, a otros los conocí en el transcurso de la misma, pero sin importar como hemos formado un bonito grupo de amigos, a ellos quiero decirles gracias por brindarme su amistad.

Finalmente quiero agradecer a los docentes de la carrera quienes inculcaron sus conocimientos para formarme profesionalmente. En especial quiero agradecer al Ing. Marco Pilatasig por darnos la oportunidad de desarrollar este proyecto de investigación que nos ayudó a afianzar nuestros conocimientos, al Ing. Edwin Pruna por brindarnos consejos que nos sirvió para sacar adelante este trabajo y al Ing. Franklin Silva que de igual manera nos brindó recomendaciones que fueron útiles en el desarrollo de nuestro trabajo final.

John Espinoza

ÍNDICE DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

En este proyecto de titulación se presenta la implementación de dos algoritmos de control avanzado en tarjetas embebidas de bajo costo, para así obtener resultados experimentales en una planta de flujo de caudal. En primera instancia se desarrolla un controlador borroso que consta de 7 funciones de membresía y opera sobre un sistema SISO (simple input / simple output), cuya entrada es el valor del error acumulado producto de la diferencia entre el valor de Set-point deseado y el valor de la variable de proceso y cuya salida actúa sobre el variador de frecuencia propio del proceso. Como segundo controlador se ha implementado un MPC (Control Predictivo Basado en Modelo), el cual para su desarrollo requiere el modelo matemático experimental del proceso ya sea discreto o continuo, por lo cual dicho modelo fue obtenido mediante el software MATLAB. Dichos controladores son implementados en las tarjetas embebidas: Arduino Uno, Raspberry Pi 3, BeagleBone Black y Udoo Neo Full, dependiendo de si las características técnicas lo permiten, la primera tarjeta utiliza código de programación abierto en C++ y las demás distintas distribuciones del Sistema Operativo Linux. Los resultados de este trabajo cuentan con una fuerte sustentación teórica, el contraste del performance de cada una de las tarjetas utilizadas y análisis de datos tomados en tiempo real mediante un HMI desarrollado en el software LabVIEW para el primer controlador y una interfaz GUI en MATLAB para el segundo. Para facilitar futuros trabajos relacionados se adjunta el código realizado, diagramas y componentes electrónicos que fueron utilizados.

PALABRAS CLAVE:

- **CONTROL AUTOMÁTICO DE PROCESOS**
- **SISTEMA DE CAUDAL**
- **TARJETAS EMBEBIDAS**
- **HMI**

ABSTRACT

In this titling project the implementation of advanced control algorithms in low-cost embedded cards is presented to obtain experimental results in a flow plant. In the first instance a fuzzy controller is developed that consists of 7 membership and opera functions on a SISO system (simple input / simple output), whose input is the cumulative error value product of the difference between the desired Set-point value and the value of the process variable and its output depend on the frequency inverter of the process. As a second controller, an MPC (Model-Based Predictive Control) has been implemented, which for its development requires the experimental mathematical model of the process and the discrete or continuous sea, so that this model was obtained through the MATLAB software. These controllers are implemented in the embedded cards: Arduino Uno, Raspberry Pi 3, BeagleBone Black and Udoo Neo Full, depending on the technical characteristics allow it, the first card uses open programming code in C ++ and the other possible distributions of the Operating System Linux The results of this work have a strong theoretical support, the performance contrast of each of the cards and the use of data taken in real time by means of an HMI developed in the LabVIEW software for the first controller and a GUI interface in MATLAB for the second To facilitate future work related to the code made, diagrams and electronic components that were used.

KEY WORDS:

- **AUTOMATIC PROCESS CONTROL**
- **FLOW SYSTEMS**
- **EMBEDDED BOARDS**
- **HMI**

CAPÍTULO I

1. INTRODUCCIÓN

1.1. Planteamiento del problema

Desde el inicio de su desarrollo en los años 30, los controladores clásicos han sido utilizados en la industria para solventar las problemáticas que han surgido en el control de procesos y sus aplicaciones, especialmente cuando la dinámica de los mismos lo han permitido. Su aplicación en las fábricas es mayor al 90% a nivel mundial; donde la mayoría de los lazos de control son del tipo PID y de éstos un gran número son del tipo PI, es decir predomina el interés del operario por utilizar algoritmos de control más sencillos, obteniendo en algunos casos un performance más modesto (por lo general restringido por las características técnicas propias de estos controladores). Hay que recalcar que la mayoría de procesos son sistemas del tipo SISO (simple entrada y simple salida), pero cuando el número de entradas y salidas aumenta o el orden del mismo es más elevado es necesaria la implementación de un controlador avanzado que brinde un mejor desempeño.

El desarrollo actual de la tecnología en el campo de la investigación permite que el diseño e implementación de controladores avanzados vaya tomando fuerza y se convierta en un motor de progreso en el desarrollo de un país, a pesar de que en el Ecuador es un término reciente, la automatización de varios procesos y la necesidad de brindar un control más robusto y estable en algunos de ellos hace que la utilización de los mismos sea ineludible.

En la industria, la mayor parte de empresas a la hora de brindar soluciones de ingeniería, opta por utilizar un PLC para el control de sus procesos, ya que hay una gran variedad de marcas y modelos, que se han ido desarrollando y mejorando a lo largo de varios años por distintos fabricantes

y para sustentar dicha inversión comercializan estos productos a precios bastante elevados, donde mientras mayor es la capacidad de entradas y salidas, la confiabilidad, las seguridades y otros factores como accesorios o la distribución del software propietario para el desarrollo de un HMI, lo van encareciendo aún más.

Esto produce que empresas manufactureras y asociaciones (de sectores varios) busquen tecnologías que brinden una solución estable a sus necesidades a costos menores. Dentro del campo de la educación, se busca brindarle al estudiante las herramientas necesarias para mejorar el proceso de enseñanza-aprendizaje, pero el costo de dispositivos como PLCs y tarjetas DAQ son cada vez más elevados, por lo cual dicho proceso muchas veces se lo ve limitado, debido a la cantidad de estudiantes que realizan las prácticas en un laboratorio. En este contexto, nace el requerimiento de evaluar las tecnologías de bajo costo que actualmente se ofrecen y determinar si son capaces de soportar algoritmos de control avanzado que puedan ser aplicadas tanto en el campo laboral como en el académico.

1.2. Antecedentes

Durante el siglo XX las técnicas para el análisis de los sistemas de control que un especialista en esta rama poseía eran el uso de ecuaciones diferenciales ordinarias (EDO), acoplado a los criterios algebraicos que identificaban la condición de las raíces de la ecuación característica asociada, lo que suponía un arduo trabajo. Luego de varios estudios aparece la reconocida Teoría Clásica de Control, que empleaba las Transformadas de Laplace y Fourier como herramientas matemáticas (Piedrafita Moreno, 1999).

Al culminar la década de los 30, se incluyó la acción derivativa a los controladores neumáticos que había en la época, con lo cual se dio origen al controlador Proporcional, Integral y Derivativo (PID), cuya sintonización empírica se dio gracias a las fórmulas que, en 1942, los ingenieros de Taylor Instruments: John G. Ziegler y Nathaniel B. Nichols introdujeron a través de

su artículo “Optimum Settings for Automatic Controllers" (Thaler, 1974). Dichas reglas para el ajuste de este tipo de controladores, aún son utilizadas y poseen gran aceptación, ya que proporcionan resultados eficaces con una mínima inversión de tiempo y cálculos (Piedrafita Moreno, 1999).

La idea inicial de control avanzado parte con la concepción de controlador autoajustable propuesto por Rudolf E. Kalman, mediante el método de reconocimiento de mínimos cuadrados recursivos del cual habla en su estudio titulado "Design of a self-optimizing control system" (Kalman, 1958) y es con el desarrollo tecnológico que actualmente cuando se hace referencia a “Control Avanzado”, se pretende mostrar el manejo de estrategias de control automático que se extienden por sobre las que comúnmente se utiliza en el control de procesos convencional (Fernández, 2006).

Posteriormente, al culminar la década de los 60, llega la era de la revolución digital, donde los nuevos dispositivos facilitan la ejecución de algoritmos de control complejos. Uno de estos dispositivos y que mayor relevancia posee es el PLC (Programmable Logic Controller), que permite la automatización de diferentes procesos, pero a costos relativamente elevados y es por ello, que de a poco varios fabricantes de tarjetas embebidas han ido desarrollando sus soluciones reduciendo el costo de inversión, pero ofreciendo menores prestaciones.

1.3. Justificación e Importancia

El estudio que se va a realizar permitirá determinar aquellas tecnologías consideradas de bajo costo, en las cuales dejando de lado el control clásico se puede desarrollar un control avanzado y aplicarlo a un proceso industrial, contribuyendo así al desarrollo de la investigación; ya que dicho proceso de control comprende una estructura completa de ingeniería que integra componentes de diversas especialidades desde “la ingeniería de control, procesamiento de señales, estadística, teorías de decisión, ingeniería de software, hasta técnicas de inteligencia artificial” (Vaca & Curay, 2015).

Dentro del proceso de control se inicia con la adquisición de datos provenientes de los sensores, se analizan los mismos, se determina el comportamiento del proceso, su dinámica, perturbaciones, posibles problemáticas y dependiendo de su performance poder perfeccionarlo de la mejor manera, realizando poca inversión en contraste al uso de PLC's y tecnologías de mayor costo.

Los resultados obtenidos en el presente proyecto servirán como base para que se propongan diversas aplicaciones y soluciones de ingeniería enfocadas a la academia y a la industria, cuyos presupuestos económicos sean limitados. Además, contribuir en el proceso de enseñanza – aprendizaje del estudiante, así como en el incremento de sus capacidades y habilidades para el manejo de tecnologías de bajo costo.

1.4. Objetivos

1.4.1. Objetivo General

- Analizar las tecnologías de bajo costo para su uso en la implementación de control avanzado en un proceso industrial.

1.4.2. Objetivos Específicos

- Investigar sobre las tecnologías de bajo costo y controles avanzados que se usan en el control automático de procesos industriales.
- Implementar un algoritmo de control avanzado usando tecnologías de bajo costo.
- Analizar las curvas de respuesta obtenidas de las diferentes tecnologías implementadas, así como las principales características técnicas y prestaciones que cada una ofrece.
- Determinar cuál de las tecnologías usadas presenta un mejor desempeño con el control avanzado implementado.

1.5. Variables de la investigación

1.5.1. Variable Independiente

Tecnologías de bajo costo.

1.5.2. Variable Dependiente

Implementación de control avanzado en un proceso industrial.

1.6. Hipótesis

El análisis de tecnologías de bajo costo permitirá determinar cuál es el sistema más óptimo para implementar un control avanzado en un proceso industrial.

1.7. Estructura del documento

En el capítulo I de este documento se brinda la descripción del problema y la importancia del proyecto, acompañado de sus respectivos objetivos e hipótesis. En el capítulo II se dispondrá el estado del arte acompañado del marco teórico respectivo. En el capítulo III se describe el diseño de los algoritmos de control. En el capítulo IV se muestra su implementación, desarrollo de las placas electrónicas y los respectivos HMIs para la muestra de datos. En el capítulo V se describen los ensayos y el análisis de los resultados obtenidos al realizar pruebas experimentales. En el capítulo VI se presentan las conclusiones y recomendaciones, con la descripción de trabajos futuros. Al finalizar el último capítulo se presenta la sección de anexos en donde se adjunta el código desarrollado en los algoritmos elaborados.

CAPÍTULO II

2. MARCO TEÓRICO

2.1. Antecedentes Investigativos

El desarrollo y continuo mejoramiento de tarjetas embebidas ha producido que los desarrolladores de software brinden mayores recursos y complementos para que se puedan realizar aplicaciones de un mayor grado de complejidad. En un inicio solo eran utilizadas para realizar control por medio de entradas y salidas digitales, ahora se ha visto su inmersión en diversos campos y en este caso particular el control de procesos.

En (Chabni, Taleb, Bembouali, & Bouthiba, 2016) se muestra la utilización de Arduino Mega 2560 acoplado a un convertidor DC/DC para controlar la variable nivel en un tanque didáctico; aplicando un controlador PID y luego uno borroso (fuzzy), donde se determina que el segundo ofrece resultados superiores. Por su parte, (Herrera Aristizábal & Hincapié Correa, 2016) implementan los mismos algoritmos de control presentados en el caso anterior pero ahora en la tarjeta Raspberry pi, para evaluar su desempeño controlando la velocidad de un motor DC; al final de su trabajo afirman que “la metodología con mejor comportamiento es la difusa ya que, además de proporcionar la respuesta con mayor rapidez, se logra una estimación más precisa que la presentada por el PID”. En (Chacón Galarza & Tapia Tapia, 2017), mediante el uso de la tarjeta embebida Beaglebone Black se ejecutan un algoritmo de control clásico y un avanzado, aplicados a una estación de flujo de aire de temperatura diseñada para el entorno académico; donde luego de realizar pruebas experimentales se determina que dicha tarjeta permite una solución industrial económica por la utilización de software libre pero que a la vez, tiene un tiempo de procesamiento lento por características propias de la misma.

También se comprueba la aplicabilidad de otros tipos de control avanzado en estas tarjetas como se detalla en (Hentzelt, Klingler, & Graichen, 2014), donde se utilizan cuatro microcomputadoras Raspberry Pi conectadas a través de una red Ethernet para la implementación de un esquema de control predictivo de modelo distribuido (DMCP) aplicado a un sistema de distribución de agua a escala en un laboratorio; brindando resultados competitivos con la utilización de un MPC (Control Predictivo por Modelo), a pesar de utilizar una red estándar y conexiones de 100 Mbps se evidenció la presencia de grandes retardos producto de las limitaciones técnicas de las tarjetas. Además se han desarrollado más trabajos relacionados como se evidencia en (Aftab, Chen, Chau, & Rahwan, 2017), donde se desarrolla un sistema para automatizar edificios aprovechando la potencia y el bajo costo de Raspberry Pi que tiene cargada un controlador MPC; su evaluación se hizo en el interior público de una mezquita, con un reconocimiento de ocupación en tiempo real cercano al 90% con un elevado ahorro de energía.

Como se ha evidenciado este tipo de tarjetas pueden ser utilizadas en aplicaciones de una mayor complejidad, brindando resultados de alto impacto y reduciendo considerablemente costos de implementación, Además se ha percibido que es un tema de investigación relativamente nuevo con publicaciones a partir del año 2014 y que en la mayoría de trabajos se realiza un contraste del rendimiento de dos algoritmos de control (uno clásico y otro avanzado) en una misma tarjeta, lo cual en este proyecto varía, al aumentar el número de dispositivos y la aplicación netamente de control avanzado.

2.2. Fundamentación Teórica

En este capítulo se presenta información de control avanzado, tarjetas de bajo costo, circuitos electrónicos y la teoría de control necesaria para el diseño de los algoritmos tanto del controlador borroso como del controlador predictivo por modelo que se han utilizado. Las descripciones se muestran a continuación.

2.2.1. Tecnologías de bajo costo

Es el nombre con el que se conoce a las tarjetas embebidas de desarrollo que varios fabricantes y makers han sacado al mercado, un fenómeno cuyo auge ha incrementado en los últimos años, dado el constante interés en brindar una mayor variedad de alternativas dependiendo de la economía y necesidades del usuario (Quirarte, 2014), a continuación, se hablará de algunas de ellas:

2.2.1.1. Arduino UNO R3

La tarjeta Arduino Uno es el producto más conocido y vendido de esta familia, es catalogado como su ícono, ya que fue elegido para el lanzamiento del software Arduino (IDE) 1.0. Está formado por elementos básicos, lo cual lo convierte en la mejor opción para iniciar en este entorno. Posee 14 pines digitales de entrada/salida, 6 de ellos son utilizados como señal PWM y 6 pines de entrada analógica, en la Figura 1 se muestra la distribución de pines de esta tarjeta y en la Tabla 1, sus principales especificaciones técnicas:

Tabla 1.
Especificaciones técnicas de la placa Arduino UNO R3

COMPONENTE	DETALLE
Procesador	Microcontrolador ATmega328P
Voltaje de operación	3.3 - 5 V
Voltaje de entrada	Enchufe de poder 6-20 V
Memoria	Flash: 32 KB SRAM: 2 KB EEPROM: 1KB
USB	1 conector USB 2.0. Puerto tipo B
Puertos seriales	1 puerto UART (también llamado USART)
Otras interfaces	1 interfaz I^2C 1 interfaz SPI
ADC interno	5 V a 10 bits

Fuente: (Kurniawan A., 2015)

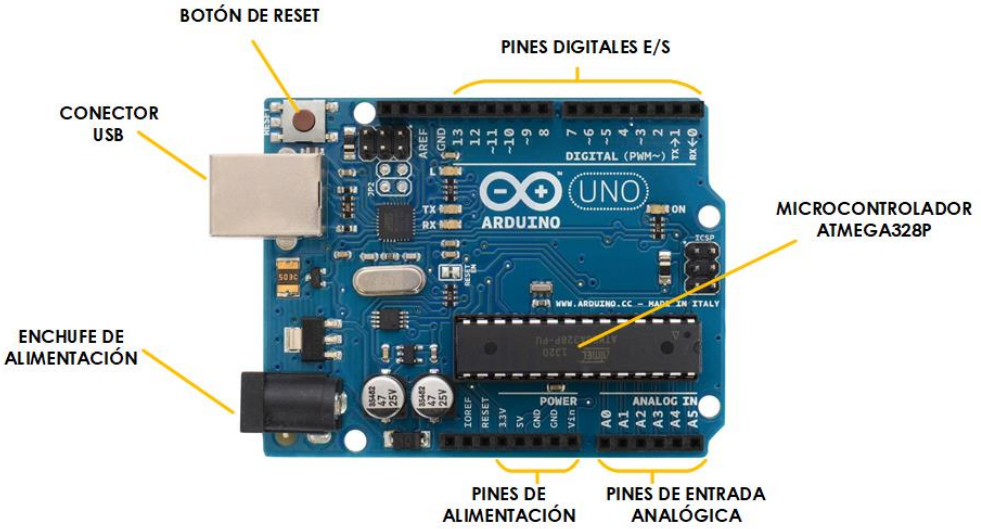


Figura 1. Arduino UNO R3

2.2.1.2. Raspberry pi 3 modelo B

Es una tarjeta de bajo costo, la tercera generación de esta familia. es un computador de placa simple (SBC) con un poderoso procesador más rápido que el de sus predecesores (10 veces mayor a la primera generación), posee 40 pines GPIO (General Purpose Input/Output), de los cuales 26 pueden ser utilizados como una entrada digital o una salida digital y operan a 3.3 V o 5 V; pero los demás son utilizados para alimentación y para otros puertos adicionales, como se muestra en la Figura 2. Posee un mayor número de dispositivos USB 2.0 (4) que versiones anteriores, donde se pueden colocar los periféricos respectivos y así trabajar como si se tratara de una computadora personal, en la Tabla 2 se muestran sus principales características (Kurniawan A. , 2016) (RS-COMPONENTS, s.f.).

Tabla 2.
Especificaciones técnicas de la Raspberry Pi 3 modelo B

COMPONENTE	DETALLE
Procesador	Broadcom BCM2387 64-bit ARMv8 - Quad-Core Cortex-A53 a 1.2 GHz
Motor de gráficos	Co-procesador Multimedia Dual Core VideoCore IV®. Proporciona Open GL

Continúa

	ES 2.0, OpenVG acelerado por hardware y decodificación de alto perfil H.264 1080p30
Voltaje de operación	3.3 - 5 V
Voltaje de entrada	Micro USB 5.1 V DC a 2.5 A
Memoria	1GB LPDDR2
Almacenamiento masivo	Ranura para tarjeta MicroSD. Interfaz SDIO de de 8-bit
Salida de video	1 interfaz HDMI (PAL y NTSC)
Salida de audio	Conector de audio minijack HDMI
USB	4 puertos USB 2.0. Puertos tipo A
Redes	1 conector RJ45 Fast ethernet 10/100 Mbps Wi-Fi 802.11 b/g/n Bluetooth 4.1 (clásico y de bajo consumo)
Puertos seriales	2 puertos UART
Otras interfaces	1 interfaz I2C 2 interfaces SPI

Fuente: (RS-COMPONENTS, s.f.)

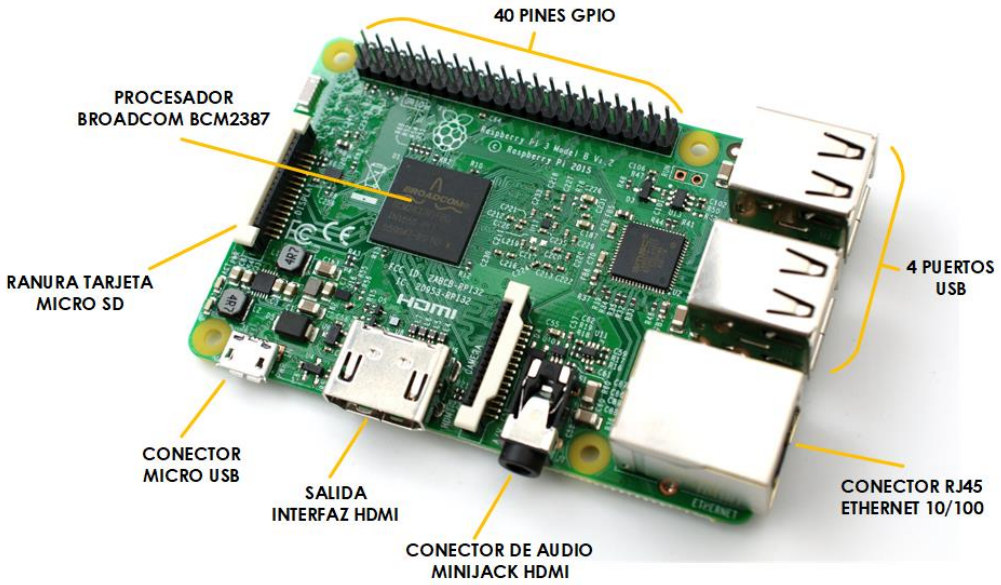


Figura 2. Raspberry Pi 3 modelo B

2.2.1.3. BeagleBone Black

Es una tarjeta embebida de bajo costo del tamaño de una tarjeta de crédito, esta plataforma de desarrollo presenta un eficiente soporte técnico y documentación relacionada que se puede encontrar en la creciente comunidad para desarrolladores y aficionados, sus especificaciones se detallan en la Tabla 3. Es perfecta para realizar pequeñas aplicaciones integradas, pues trabaja con el sistema operativo Linux, que puede arrancar en menos de 10 segundos y se puede iniciar a desarrollar en menos de 5 minutos. En la Figura 3 se pueden visualizar sus 2 cabeceras de 46 pines cada una y demás elementos (TEXAS INSTRUMENTS, 2014).

Tabla 3.
Especificaciones técnicas de la tarjeta Beaglebone Black

COMPONENTE	DETALLE
Procesador	SITARA AM335x a 1 GHz ARM Cortex-A8
Motor de gráficos	SGX530 3D, 20 M Polygons/S
Voltaje de operación	3.3 - 5 V
Voltaje de entrada	Mini USB 5 V DC Enchufe de poder 6-15 V
Memoria	512 MB DDR3L 800 MHz
Almacenamiento masivo	Ranura para tarjeta MicroSD. Interfaz de 8-bit eMMC
Salida de video	16b HDMI, 1280X1024 (MAX)
Salida de audio	1 transmisor de audio stereo HDMI
USB	1 puerto USB 2.0. 1 enchufe tipo A
Redes	1 conector RJ45 Fast ethernet 10/100 Mbps
Puertos seriales	1 puerto UART
Otras interfaces	1 interfaz I2C 1 interfaz SPI
ADC interno	1.8 V a 12 bits

Fuente: (Linux, 2017)

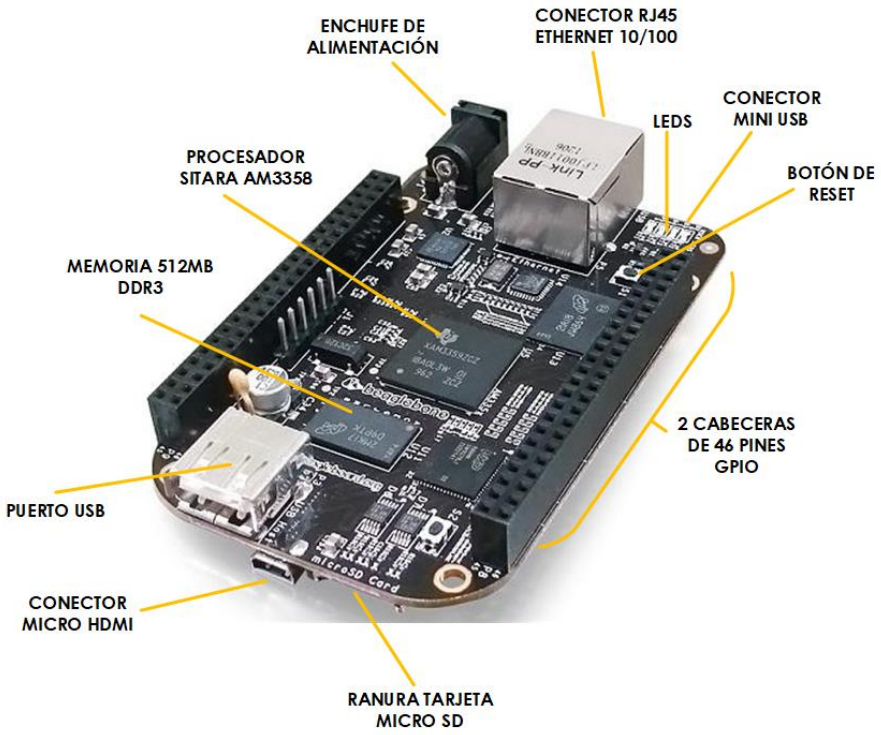


Figura 3. Beaglebone Black

2.2.1.4. UDOO neo full

Es un computador de placa única embebida que opera con Android/Linux y adhiere su propia versión de Arduino (IDE). Varios buses SPI, I^2C y UART (aunque la mayoría de estos buses vienen dados para componentes propios de la tarjeta y no se pueden usar) como se muestra en la Figura 4. La principal característica de esta tarjeta es su procesador heterogéneo, con dos núcleos integrados en un solo procesador NXP i.MX 6SoloX. El primero es ARM Cortex-A9 a 1 GHz, que posee una plataforma compatible con Arduino UNO, ya que además posee su misma cabecera de pines, con la diferencia que su ADC interno es de 3.3 V solamente. El segundo co-procesador es Cortex-M4 de E/S en tiempo real, las demás características técnicas se muestran en la Tabla 4.

Se lo puede utilizar como un computador personal de baja potencia al conectarlo a una pantalla mediante un cable micro HDMI a HDMI y utilizando su único puerto USB conectar un mouse, para el caso del teclado se incorpora

uno en el escritorio por defecto. En otras ocasiones se lo puede utilizar para realizar diversas aplicaciones que quedan a libertador del desarrollador, como ejemplos: construcción de un dron, robots, impresora 3D básicas, como parte de un sistema de domótica para la automatización del hogar. (UDOO, 2016).

Tabla 4.
Especificaciones técnicas de la tarjeta UDOO Neo Full

COMPONENTE	DETALLE
Procesador	NXP i.MX 6SoloX
Motor de gráficos	Vivante GC420, acelerador gráfico integrado 2D/3D.
Voltaje de operación	3.3 - 5 V
Voltaje de entrada	Micro USB 5 V DC Enchufe de poder 6-15 V
Memoria	1 GB
Almacenamiento masivo	Ranura para tarjeta MicroSD. Interfaz SDIO de 8-bit
Salida de video	1 interfaz Micro HMI 1 interfaz LVDS + táctil (señales I2C)
Salida de audio	1 transmisor de audio HDMI 1 S/PIDF & I2S
USB	1 puerto USB 2.0. Puertos tipo A 1 puerto USB OTG (conector micro-AB)
Redes	1 conector RJ45 Fast ethernet 10/100 Mbps Wi-Fi 802.11 b/g/n Bluetooth 4.0 Baja energía
Puertos seriales	3 puertos UART
Otras interfaces	3 interfaces I2C 1 interfaz SPI
Sensores integrados	1 acelerómetro de 3 ejes 1 magnetómetro de 3 ejes 1 giroscopio digita de 3 ejes
ADC interno	3.3 V a 12 bits

Fuente: (UDOO, 2016)

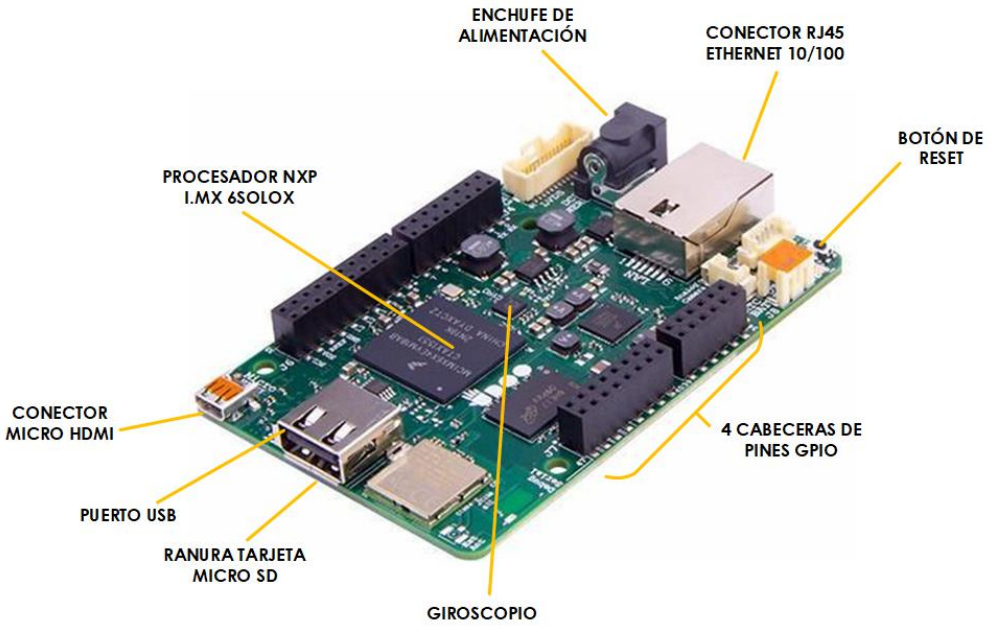


Figura 4. Tarjeta UDOO Neo Full

2.2.2. Software

2.2.2.1. KEPserverEx 5

Es una plataforma de conectividad que permite al usuario enlazar, administrar, controlar y monitorear varios dispositivos de automatización y aplicaciones de software. La plataforma KEPServerEX 5 aprovecha la tecnología OPC de la interoperabilidad, la escalabilidad y los protocolos de comunicación para proporcionar a los usuarios una fuente unificada de datos industriales como se presenta en la Figura 5. (Kepware, 2017)

Características:

- Permite accesibilidad a los datos de varias aplicaciones.
- Optimiza las comunicaciones.
- Posee una amplia conectividad con varios controladores.
- Ofrece seguridad en las comunicaciones.
- Ofrece la posibilidad de diagnóstico de las comunicaciones.

Figura 5. Adquisición de datos centralizados con KEPserverEX

Fuente: (Kepware, 2017)

2.2.2.2. Ubuntu Mate 16.04

Es un sistema operativo de la distribución de Linux para Raspberry Pi 2 y Raspberry Pi 3 que posee un entorno configurable, estable y sencillo de manejar (véase la Figura 6); además se caracteriza por su ligereza a la hora de ejecutarse por los pocos recursos de hardware que requiere.

Prestaciones:

- Rendimiento optimizado al agregar redimensionamiento automático de la partición de arranque.
- Deshabilita servicios redundantes para disminuir los ciclos de CPU y requerimientos de RAM.
- En la última versión Ubuntu MATE 16.04 presenta reproducción de video acelerado por hardware en VLC y decodificación acelerado por hardware y codificación en ffmpeg. (Ubuntu, 2017)

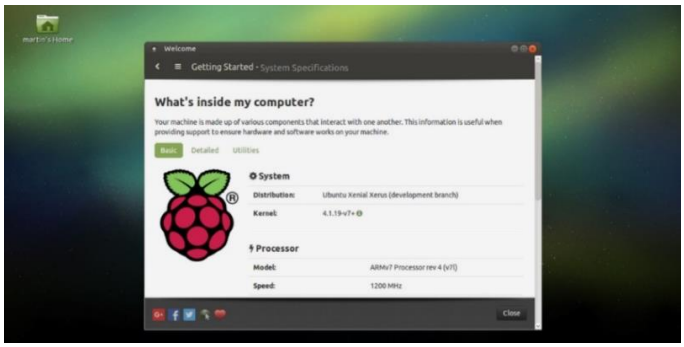


Figura 6.Escritorio de Ubuntu MATE en la Raspberry Pi3

Fuente: (Ubuntu, 2017)

2.2.2.3. Debian

Es un sistema operativo libre adaptada para diferentes núcleos como Linux, Hurd, NetBSD y kFreeBSD, siendo la más desarrollada Debian GNU/Linux. En la actualidad Debian adoptó una imagen del sistema operativo para Beaglebone Black con varios beneficios en paquetes y softwares compatibles. (véase la Figura 7)

Figura 7. Sistema Operativo Debian

Fuente: (Torres, 2015)

Características:

- Puede aplicarse en varias arquitecturas ya que contiene soporte para 12 plataformas.
- Posee una extensa colección de software (más de 51000 paquetes).
- Posee un amplio soporte tanto en su instalación, actualización y uso de sus herramientas.
- No tiene un entorno gráfico específico, por lo que el usuario tiene la facilidad de elegir e instalar uno de su interés, entre estos está GNOME, KDE, Xfce, LXDE u otro. (Debian, 2017).

2.2.2.4. MATLAB

Es un entorno de programación con altas prestaciones técnicas, destinada al cálculo numérico y visualización. Incorpora en su plataforma:

- Análisis numérico
- Cálculo matricial
- Procesamiento de señales
- Procesamiento gráfico

MATLAB es una herramienta de programación ampliamente usada para resolver problemas de ingeniería y matemáticas, dando importancia a las aplicaciones de control y procesamiento de señales. Para proporcionar estas soluciones el software facilita una serie de toolboxes y herramientas adicionales como por ejemplo Control System Toolbox, Simulink y GUIDE, las cuales se encuentran clasificadas en las siguientes categorías (EcuRed, 2017), véase la Tabla 5:

Tabla 5.
Tabla de categorías de MATLAB Y Simulink

MATLAB (Toolboxes)	Simulink
Matemáticas y Optimización	Modelado de punto fijo
Estadísticas y Análisis de datos	Modelado basado en eventos
Diseño de sistemas de control y análisis	Modelado físico
Procesado de señal y comunicaciones	Gráficos de simulación
Procesado de imagen	Diseño de sistemas de control y análisis
Pruebas y medidas	Procesado de señal y comunicaciones
Biología computacional	Generación de código
Modelado y análisis financiero	Prototipos de control rápido y SW/HW HIL
Desarrollo de aplicaciones	Tarjetas integradas
Informes y conexión a bases de datos	Verificación, validación y comprobación
Compiler	Verificación, validación del código y desarrollo de ejecutables.

Fuente: (EcuRed, 2017)

2.2.2.5. Simulink

Es un software destinado al modelado, simulación y análisis de sistemas dinámicos; además proporciona un editor gráfico y bibliotecas cada una con bloques de trabajo personalizables. Como se menciona en el apartado anterior está integrado con MATLAB, con esta facilidad permite incorporar algoritmos desarrollados en MATLAB y exportar posteriormente los resultados de simulación a MATLAB para su análisis (Matlab, 2015).

Características:

- Un editor gráfico para construir esquemas de bloques jerárquicos.
- Diferentes bloques para modelar sistemas en tiempo discreto y continuo.
- Pantallas de simulación en tiempo real.
- Herramientas de análisis de modelos para aumentar la velocidad de simulación
- Legacy Code Tool para importar código C y C++ en los modelos de simulación. (MathWorks, Inc, 2015)

2.2.2.6. Model Predictive Control Toolbox

El toolbox de control predictivo de Matlab permite diseñar un control óptimo sobre una planta combinando una estrategia de predicción y de control. El toolbox permite usar las funciones de transferencia, matrices de estado, restricciones e incluso retrasos que permitan simular de la forma más real el rendimiento del controlador en las plantas industriales. Si no se posee un modelo de la planta se puede usar el toolbox para identificación de sistemas y desarrollar un modelo basado en los datos, véase la Figura 8 (MathWorks, Inc, 2017)

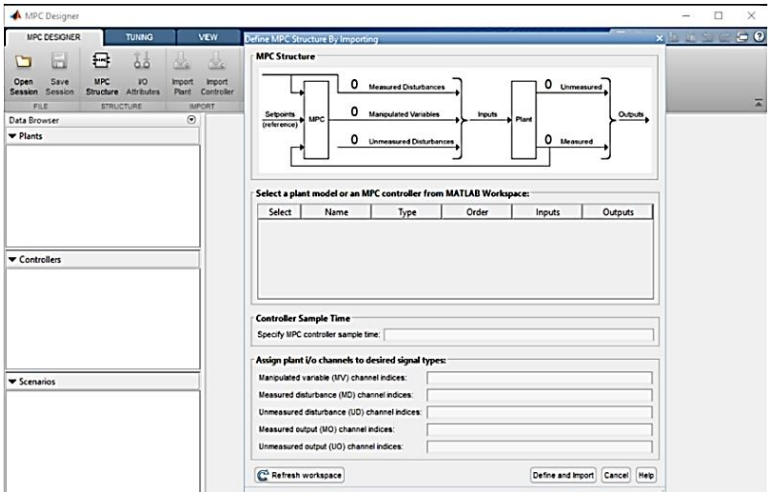


Figura 8. **Toolbox del control predictivo basado en el modelo**

2.2.2.7. LabVIEW 2014

Es un software especializado en informática industrial y científica; que facilita la integración de cualquier dispositivo de E/S (véase Figura 9). Su uso es relativamente sencillo ya que basa su lenguaje de programación de forma gráfica, reduciendo significativamente el tiempo de desarrollo. Posee librerías para la adquisición de datos, el control de procesos, análisis matemático, comunicación con hardware GPIB, VXI, RS-232, RS-485, entre otras. En la Tabla 6 se puede observar las ventajas de usar LabVIEW (Carvajal & Proaño, 2015).

Tabla 6.
Ventajas de usar LabVIEW

Programación gráfica intuitiva	Posee herramientas que se adaptan al usuario solo con la visualización y modelización.
Herramientas de depuración interactiva	Al poseer un entorno de programación gráfico, el depurado se vuelve más rápido e intuitivo.
Paralelismo y Rendimiento automático	LabVIEW permite ejecutar de forma paralela partes el código.
Combinación de la programación G con otros lenguajes	Posibilidad de elegir una programación gráfica, textual o mixta.

Fuente: (National Instruments, 2015)

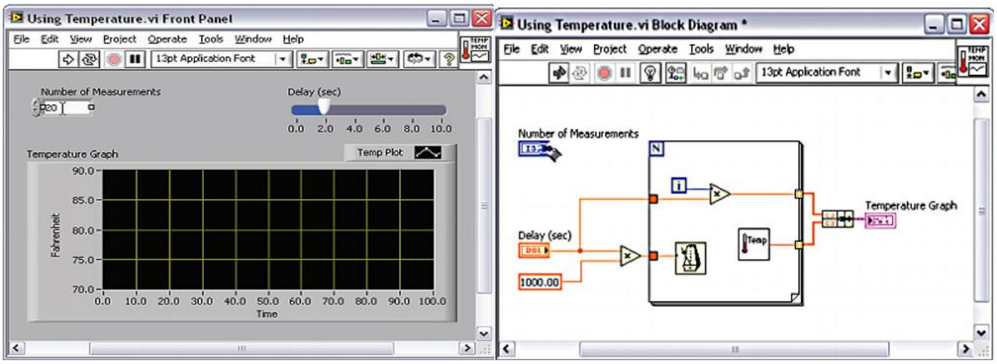


Figura 9. Software LabVIEW
Fuente: (National Instruments, 2015)

2.2.2.8. Proteus 8 Professional

Es un software creado por Labcenter Electronics usado para en el desarrollo de proyectos electrónicos en las etapas de: diseño, simulación, corrección de errores, elaboración de la placa impresa y construcción. Está formado por dos programas principales: Isis, que nos permite diseñar y simular cualquier circuito y Ares, el cual nos permite elaborar el circuito impreso del proyecto desarrollado con la ventaja de obtener una vista previa en tres dimensiones de nuestro proyecto (HUBOR, 2015).

Ventajas:

- Al simular reduce gastos, ya que permite realizar pruebas del diseño sin correr riesgos.
- Optimiza el tiempo, dado que la implementación será más rápida una vez comprobado su correcto funcionamiento por simulación.
- Se puede encontrar errores y problemas de diseño de una manera más fácil.

Desventajas:

- Solo está disponible para el Sistema Operativo de Windows.
- Es un software propietario.

En la Figura 10 se puede observar la optimización de tiempo al usar Proteus como herramienta de simulación previa. (HUBOR, 2015)

Figura 10. Fases de diseño al usar Proteus

Fuente: (HUBOR, 2015)

2.2.3. Sistemas de Control

Los sistemas de control han estado presentes en muchos ámbitos de la vida del ser humano, como lo son: alimentación, educación, transporte, entretenimiento, investigación, salud, etc. Y es a partir de la década de los años 50 que su importancia se torna preponderante ya que son un pilar trascendental del desarrollo de la tecnología y a la vez su promotor. Según la necesidad, se definen las estrategias de control que serán utilizadas.

Un sistema de control es una agrupación de componentes que han sido delineados, con la finalidad de autorregular su comportamiento o el de otro sistema; para realizar una determinada tarea, adaptándose a las condiciones ambientales y exigencias del usuario (Castiñeira, s.f.). Para ello se debe medir el valor de la variable de proceso y aplicar una acción de control en la variable manipulada para así llegar a un valor deseado.

Un sistema de control es un arreglo de componentes físicos diseñados, de tal manera que se pueda manipular, dirigir o regular a sí mismo o a otro sistema, a través de una acción de control (Dulhoste, s.f.). La acción de control

debe controlar al sistema; y para controlar se requiere medir el valor de la variable que se está midiendo del sistema y aplicar la variable manipulada al sistema para corregir o limitar una desviación del valor medio a partir de un valor deseado (Núñez Enríquez, 2007).

Por lo general se los clasifica en dos grandes grupos:

- **Lazo abierto.** Son sistemas no realimentados en los cuales no es necesario medir el valor de la salida para compararlo con el valor de la entrada (Dulhoste, s.f.).

- **Lazo cerrado.** Son sistemas de control realimentado, donde la entrada del controlador es una señal de error actuante (diferencia entre el valor de una entrada de referencia y la señal de salida realimentada), de esta manera se busca minimizar dicho error continuamente. En la Figura 11 se puede apreciar los componentes de este tipo de sistema (Prof. Castillo Rubio, 2008).

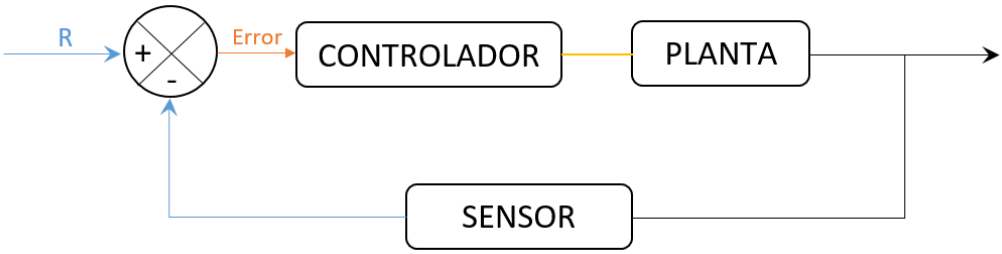


Figura 11. Diagrama de bloques de un sistema de lazo cerrado

Las principales técnicas de control se presentan brevemente a continuación:

2.2.3.1. Acción de Control Proporcional Integral Derivativa (PID)

Controlador retroalimentado que utiliza las prestaciones de las acciones que lo conforman y cuya salida está dada por la siguiente ecuación:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e(t)\partial t + K_p T_d \frac{\partial e(t)}{\partial t} \qquad (2.1)$$

La acción proporcional produce una señal de control proporcional a la señal de error que ingresa al controlador, la integral corrige el error de régimen permanente de la acción previa (desviación constante del valor de la variable de proceso con respecto al valor deseado) y la acción derivativa logra suprimir las oscilaciones (Ruiz Canales & Miguel, 2010). Cuando la señal de error va cambiando de una forma más lenta en el tiempo, predominan las acciones proporcional e integral derivativa y cuando cambia de forma opuesta actúa la acción derivativa. (E-ducativa, 2017). Con este controlador se obtiene una alta velocidad de reacción, tiempo muerto moderado y rápida respuesta a las perturbaciones, aunque es propenso a presentar oscilaciones (Miranda Medrano, 2017), en la Figura 12 se presenta su diagrama de bloques:

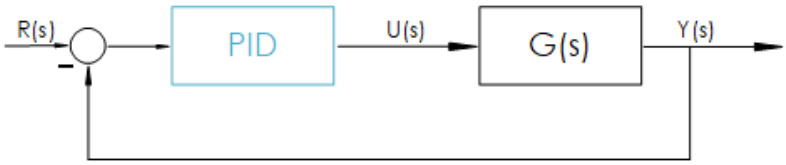


Figura 12. Diagrama de bloques de la estructura del PID

Fuente: (Mazzone, 2002)

2.2.3.2. Control avanzado

Es una infraestructura de un sistema de ingeniera que reúne un conjunto de estrategias de control que son aplicadas a un proceso y engloban una variedad de disciplinas para su diseño. Su utilidad se comprueba al aplicarlos a sistemas complejos con múltiples entradas y múltiples salidas (MIMO); a diferencia de los sistemas de una entrada y una salida (SISO) donde se utilizan controladores clásicos principalmente, que a pesar de ser fáciles de desarrollar presentan ciertas limitaciones, a pesar de eso son los más utilizados en la industria. Para el diseño de algoritmos de control avanzado se requiere que tanto hardware como software cumplan con los requerimientos mínimos del sistema, ya que su involucran una robusta plataforma computacional; además, se basan en el completo entendimiento del proceso, su fenomenología, dinámica, posibles perturbaciones y demás características técnicas. A partir del modelo matemático que permite una

aproximación del sistema se puede realizar simulaciones de su comportamiento y a la hora de comercializarse no son muy comunes en el mercado, pues su desarrollo lo realiza generalmente un experto en control. (Fernández, 2006).

A partir de su definición existe una variada clasificación, a pesar que no son muy comúnmente utilizados por su grado de dificultad, lo cual ha motivado que muchos de ellos no traspasen la barrera del ámbito académico-científico al profesional. A continuación, se menciona las estrategias de control más usadas:

2.2.3.2.1. Control experto

Se fundamenta en la colección de conocimientos acerca del sistema que pueden describirse como “expertos”. Su principal ejemplo es el control borroso (más conocido como fuzzy control), el cual convierte un conjunto de reglas del tipo “SI… ENTONCES” (IF… THEN) en consecuencias o acciones del control, su aceptación en el ámbito industrial está dado por la variedad de aplicaciones en las cuales se lo utiliza y por la facilidad con la que se ajusta a las necesidades del sistema.

2.2.3.2.2. Control óptimo

Su nombre viene dado porque se pretende dar una acción de control óptima dependiendo de los criterios previamente establecidos, que vienen dados de la obtención del modelo matemático del proceso a controlar, donde se incluye por lo general el error, acción y desviaciones de control e incluso se puede establecer limitaciones. Su principal ejemplo es el control predictivo por modelo (MPC por sus siglas en inglés), el cual trata de predecir el comportamiento futuro del sistema frente a posibles acciones de control aplicadas.

2.2.3.2.3. Redes neuronales

Los ANS (sistemas neuronales artificiales o más conocidos como redes neuronales: RN) son estructuras matemáticas que buscan teorizar la actividad del cerebro humano y trabajar como él, yendo más allá del procesamiento local y distribuido. Se utilizan para el tratamiento y comprensión de datos, reconocimiento de patrones y de imágenes, aproximaciones en general y principalmente para brindar soluciones industriales, ya que a diferencia de una computadora tradicional son capaces de realizar operaciones para las cuales no fueron previamente programadas a detalle (Redondo Fonseca, 2016).

2.2.3.2.4. Control Robusto

Abarca los problemas que se caracterizan por apreciar incertidumbres en el modelo matemático que sea admisibles por un controlador lineal y que no se modifiquen en el tiempo (Ramirez Ramos, 2008), independientemente de las perturbaciones que puedan aparecer. Sus principales representantes son el control por modelo interno (IMC) y el control mixto H_2/H_∞. (NCS, s.f.).

2.2.4. Control borroso

La teoría de lógica borrosa tiene como objetivo es brindar un sistema frontal computacionalmente dotado de técnicas para trabajar con un razonamiento aproximado, en lugar de uno exacto. Es decir, todo está en términos de grado de pertenencia a un conjunto.

Al basarse en la relatividad de lo apreciado, permite conseguir de una forma sencilla una conclusión a partir de la información de una entrada ambigua. Su principal ventaja es la facilidad con la que se adapta a las experiencias y al entorno en el que se desenvuelve una persona, pues trabaja mediante etiquetas lingüísticas como: “el nivel de agua es MUY ELEVADO” o “la temperatura del horno es BAJA”; con ello se trata de cuantificar las

expresiones verbales y darles un valor numérico entre 0 y 1 dependiendo de lo que se requiera. Un aspecto similar que se presenta en los controladores clásicos, los cuales buscan transformar un espacio de entrada en un espacio de salida, con la diferencia que estos no permiten asignar valores intermedios, solo 0 o 1.

Es una técnica multidisciplinaria con grandes aplicaciones en la actualidad, sobre todo en electrodomésticos, procesamiento de imágenes, frenos de trenes, etc. Esto se da por su sencillez conceptual al no requerir complicados algoritmos para su implementación, lo cual es su mayor ventaja en comparación a otros controladores avanzados y su respuesta rápida a perturbaciones en el proceso.

2.2.4.1. Teoría de conjuntos borrosos

Lotfi Asker Zadeh a diferencia de la teoría clásica de conjuntos, donde uno de sus elementos puede pertenecer un conjunto o puede no hacerlo; propuso que dicho elemento puede pertenecer a más de un conjunto, pero teniendo diferentes grados de pertenencia definidos como un valor numérico entre 0 y 1 como se muestra en la Figura 13.

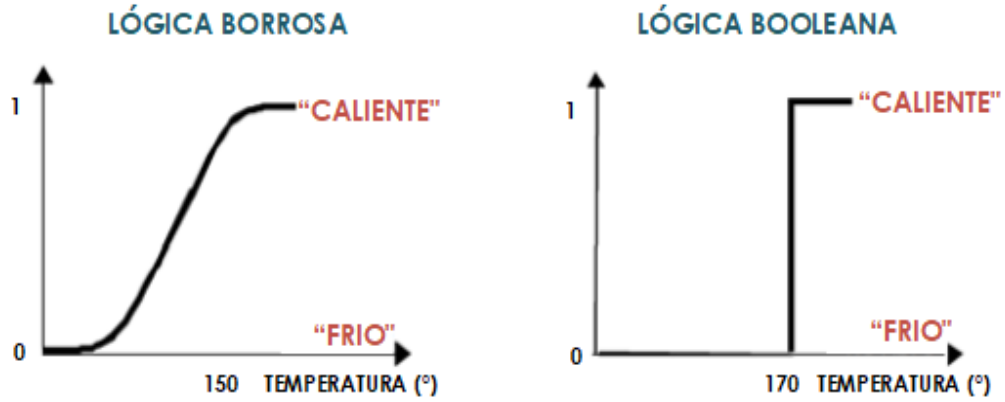


Figura 13. Representación gráfica de la teoría de conjuntos borrosos

2.2.4.2. Variables lingüísticas

Como hace referencia su nombre, son variables que en lugar de tener valores numéricos tienen sentencias formadas de términos comunes para el diseñador.

Esta lógica permite manipular información que no es precisa, como la temperatura de un horno. Así, por ejemplo, si la temperatura menor a 100° se lo considera frío, si es superior a 100° e inferior a 150° se lo considera normal y si igual o mayor a 150° se lo considera caliente. Sin embargo, la diferencia entre una temperatura de 149° y 150° es mínima, pero según la teoría clásica de conjuntos los separa en dos grupos distintos. Si en lugar de eso se adoptara la definición de conjuntos difusos, estos cambios abruptos se suprimirían, pues los límites entre conjuntos mostrarían cambios más paulatinos, como se presenta en la Figura 14.

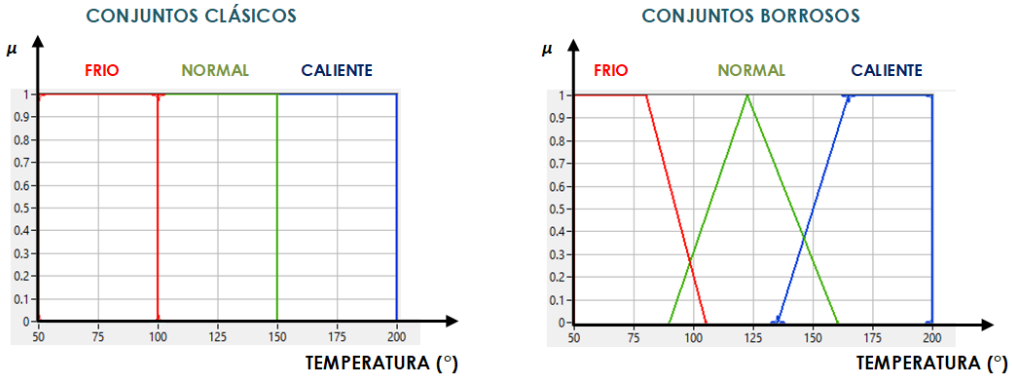


Figura 14. Conjuntos Borrosos para la variable Temperatura

2.2.4.3. Funciones de pertenencia

El grado de pertenencia se define a través de una función característica asociada a un conjunto difuso, que toma el nombre de función de membresía o de pertenencia. Es decir, si se define un conjunto “C” con n elementos: la función de pertenencia está dada $\mu_A(x)$, donde para cada valor que pueda tomar la variable de entrada x, la función de membresía brinda el grado de

pertenencia de este valor X al conjunto difuso, siempre y cuando cumple la condición:

$$\mu_A(x) \in [0,1] \tag{2.2}$$

Las funciones de membresía permiten representar de manera gráfica un conjunto borroso sobre un universo. La forma que tome dicha función depende del criterio elegido para el diseño y el universo depende del diseñador del sistema. A continuación, se muestran los principales tipos de funciones:

2.2.4.3.1. Función trapezoidal

Está definida por límites inferior a y superior b (ver Figura 15); además, los límites de soporte inferior α y superior β.

$$\mu_A(X)\begin{cases} 0 & si\ x < a \\ \dfrac{x-a}{\propto -a} & si\ a \le x < \alpha \\ 1 & si\ \alpha \le x < \beta \\ \dfrac{\beta - x}{b-\beta} & si\ \beta \le x < b \\ 0 & si\ x \ge b \end{cases}$$

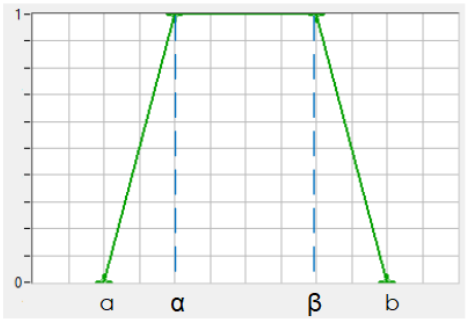


Figura 15. Representación gráfica de la Función Trapezoidal

2.2.4.3.2. Función triangular

Está definida por límites inferior a, superior b y el valor modal p (Figura 16). Cabe aclarar que la función no tiene que ser simétrica necesariamente.

$$\mu_A(X)\begin{cases} 0 & si\ x < a \\ \dfrac{x-a}{p-a} & si\ a \le x < p \\ \dfrac{b-x}{b-p} & si\ p \le x < b \\ 0 & si\ x \ge b \end{cases}$$

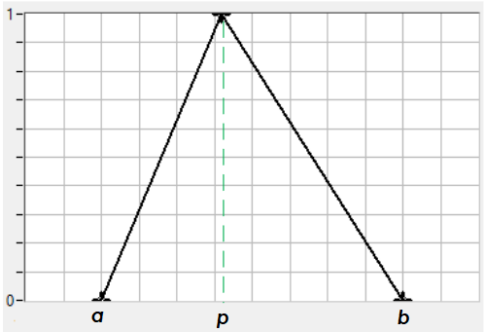


Figura 16. Representación gráfica de la Función Triangular

2.2.4.3.3. Función gaussiana

Está definida por su valor medio p y un parámetro k<0. Está dada por la función de campa de Gauss (ver Figura 17) y mientras más grande es el valor de k, más estrecha es la campana.

$$\mu_A(X) = e^{-k(x-p)^2}$$

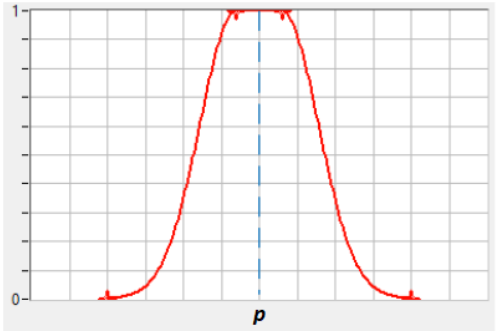


Figura 17. Representación gráfica de la Función Gaussiana

2.2.4.3.4. Función sigmoide

Está definida por límites inferior a, superior b y el valor p o punto de inflexión. El caso más usual es cuando p = (a+b)/2 como se evidencia a continuación (Figura 18):

$$U_A(X)\begin{cases} 0 & si\ x < a \\ 2\left[\frac{x-a}{b-a}\right]^2 & si\ a \le x < p \\ 1-2\left[\frac{x-b}{b-a}\right]^2 & si\ p \le x < b \\ 1 & si\ x \ge b \end{cases}$$

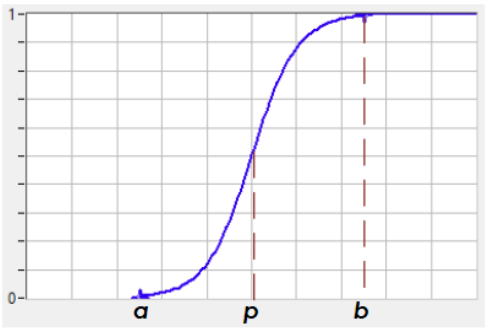


Figura 18. Representación gráfica de la Función Sigmoide

2.2.4.4. Etapas de un controlador borroso

En la Figura 19 se puede apreciar la estructura de un controlador borroso y a continuación las etapas principales del mismo.

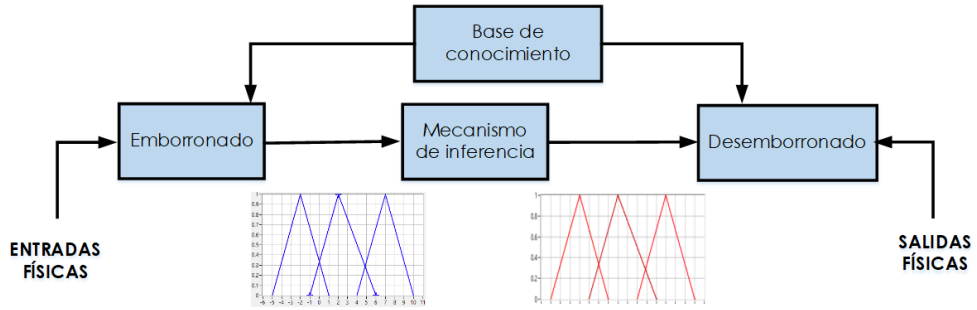


Figura 19. Diagrama de bloques de un controlador borroso

2.2.4.5. Emborronado

Más conocido como fusificación (fuzzification), es la etapa en la cual se convierten las variables físicas de un sistema en variables lingüísticas, es decir se toma cada parámetro físico y se le otorga un grado de pertenencia en el conjunto borroso que se le corresponda, mediante las funciones de membresía asociados a dichos conjuntos.

Primero se divide el universo de discurso y se asignan las etiquetas lingüísticas correspondientes con sus correspondientes valores numéricos a través de la función de membresía que le pertenece, para finalmente establecer su grado de pertenencia (Abril & Pacheco, 2012)

2.2.4.6. Base de conocimiento

Envuelve un conocimiento del dominio de aplicación y consta de dos elementos: una base de datos y una base de reglas del comportamiento del sistema. El primero engloba la definición lingüística de las variables de entrada y salida. El segundo, combina uno o más conjuntos borrosos de entrada, denominados antecedentes o premisas y los relaciona con un conjunto borroso de salida, que toma el nombre de consecuente o consecuencia. Como se muestra a continuación:

$$IF\ premisa\ (antecedente), THEN\ conclusión\ (consecuente)$$

Estos conjuntos borrosos se relacionan a través de las operaciones lógicas AND y OR. Estas reglas se van formando en base al conocimiento que se tiene sobre el sistema a controlar y dependiendo del número de entradas y salida puede ir aumentando, por lo cual se recomienda agruparlos en una tabla para un mejor entendimiento o bien en una memoria asociativa difusa (más conocida como FAM).

2.2.4.7. Inferencia lógica

Segú la base de conocimiento formada para el modelo, se determina el sistema de inferencia lógica adecuado, el cual realiza el razonamiento formal con proposiciones, que, a diferencia de la lógica booleana, puede tomar valores de las proposiciones intermedios entre verdadero y falso. Los dos tipos más usados son Mamdani y Sugeno.

2.2.4.7.1. Inferencia lógica tipo Mamdani

Su formato es de la forma:

$$\text{SI } u_1 \text{ es } A_1 \text{ Y } u_2 \text{ es } A_2 \text{ Y } \ldots \text{ Y } u_n \text{ es } A_n \text{ , ENTONCES v es B} \quad (2.3)$$

Donde los u_j y v son variables lingüísticas; y los A_j y B simbolizan los valores lingüísticos que dichas variables deben asumir. El lado izquierdo de la regla (LI) está formado por los antecedentes y el lado derecho (LD) es el consecuente. Este método es bastante instintivo y altamente aprobado por acoplarse mejor al lenguaje humano.

2.2.4.7.2. Inferencia lógica tipo Sugeno

Su formato es de la forma:

$$SI\ u_1\ es\ A_1\ Y\ u_2\ es\ A_2\ Y\ \ldots Y\ u_n\ es\ A_n, ENTONCES\ v\ es\ f(u_1, u_2, \ldots, u_n) \quad (2.4)$$

Donde los u_j son variables lingüísticas de entrada; los A_j representan los valores lingüísticos que dichas variables pueden asumir; v es la variable

de salida y f la función lineal de las entradas. Es más eficiente en términos computacionales que el anterior y trabaja mejor con técnicas de optimización y adaptativas, aunque requiere un mayor conocimiento de este tipo de control.

2.2.4.8. Desemborronado

Más conocida como defusificación (defuzzification), es la etapa en la cual en base a un conjunto difuso que se obtiene por la máquina de inferencia, mediante métodos matemáticos de defusificación lo transforma a un valor numérico. Los métodos utilizados son los siguientes:

- **Método del máximo (CoM):** Es el caso en donde la función característica del conjunto difuso tiene su máximo valor, tomando ese valor como salida.

- **Método del centroide o centro de área (CoG)**: Se calcula el centro de gravedad del área limitada por la curva de la función de membresía.

- **Media de máximos (MoM).** La salida está definida por la ecuación 2.4, en donde R\ es el valor medio de los puntos de máximo grado de pertenencia de la función de membresía.

2.2.5. Control Predictivo Basado en Modelo (MPC)

El control Predictivo basado en el modelo, es un algoritmo de control que permite optimizar el proceso de la planta, al predecir el desempeño futuro y calcular las acciones de las variables manipuladas y controladas del proceso para alcanzar los objetivos esperados. El criterio a optimizar, se lo conoce como función de coste y está vinculado con las acciones futuras del sistema dadas por el modelo de predicción (Vaca & Curay, 2015). El horizonte de predicción es el intervalo de tiempo futuro que actúa en la optimización y que

está dado por instantes de muestreo futuros. En el MPC también interviene el horizonte de control el cual debe ser menor al horizonte de predicción, para que el funcionamiento del controlador sea constante en comportamientos que superen al horizonte de control (Aguilar & Ortiz, 2017).

La estrategia de control que se usa para dar robustez al sistema es la del horizonte deslizante, que consiste en usar las acciones calculadas durante un cierto periodo de tiempo y luego volver a realizar un cálculo “online” para obtener una nueva solución óptima. De esta forma, el horizonte de predicción se va deslizando conforme el tiempo sigue avanzando (Limón Marruedo, 2002).

2.2.5.1. Control Predictivo

El control predictivo también conocido como control multivariable, es una estrategia de control que en la actualidad es una de las alternativas que mejor se adapta a procesos que poseen varias entradas y varias salidas, además que en su estructura acepta cualquier tipo de restricciones. Las estrategias de control clásicas se adaptan perfectamente a procesos industriales lineales, pero para procesos no lineales es necesario algoritmos de control óptimos y que sean sencillos de aplicar.

En la actualidad la aplicación del control MPC ha dejado de limitarse solo a la industria petrolera y se ha extendido a la manufactura química, automotriz, alimenticia, aviónica y metalúrgica. El éxito de este controlador se debe a las razones que se describen a continuación:

- Incorpora un modelo del proceso a controlar, lo que le permite manejar todas las singularidades de la dinámica del proceso.

- El algoritmo del MPC toma en cuenta el comportamiento futuro de la planta. Esta conducta significa que el controlador puede anticipar

perturbaciones y eliminarlas, permitiendo que la salida del proceso siga la trayectoria deseada.

- El diseño del algoritmo incorpora restricciones de salida que evitan violaciones al control y que mejoran significativamente el resultado final del proceso a controlar, siendo esta una de las características que más distingue al MPC de otros controles (Vaca & Curay, 2015).

2.2.5.2. Ventajas del MPC

Algunas de las ventajas que presenta el MPC son:

- Permite tratar con sistemas lineales, no lineales, monovariables y multivariables.
- Incorpora en su diseño un modelo previo y las restricciones de las variables, lo que permite que el controlador actúe con un “molde” que se asemeja a la dinámica real del sistema y maximice su rendimiento.
- Tiene en cuenta las limitaciones que posee un actuador físicamente.
- La ley de control se ajusta a los criterios óptimos.
- Son controladores que presentan una estructura flexible e intuitiva y que puede seguir creciendo en su implementación industrial.

De las ventajas que se menciona la que más sobresale es la de incorporar restricciones en su diseño previo.

2.2.5.3. Desventajas del MPC

A pesar de ser un control que mejora el rendimiento de un sistema también posee desventajas las cuales se mencionan a continuación:

- Necesita de un modelo preciso de la planta.
- La robustez del control depende directamente de un algoritmo de optimización, el cual solo puede implementarse por computador.

- El proceso de optimización necesita de un alto costo computacional, por lo que puede limitarse en la aplicación a sistemas con dinámicas muy rápidas (Limón Marruedo, 2002).

Cabe mencionar que el costo computacional se reduce notablemente si el sistema no posee restricciones o si tiene una dinámica lenta (Feroldi, 2012).

2.2.5.4. Elementos del Control Predictivo

Sin importar el tipo de proceso en el que se implementa, un control predictivo basado en el modelo resuelve problemas de optimización en cada instante de tiempo **(horizonte deslizante),** mediante 3 elementos esenciales que son:

- Modelo de predicción
- Función objetivo o Función de coste
- Ley de control

En la Figura 20 se puede observar la estructura general que tiene un controlador predictivo:

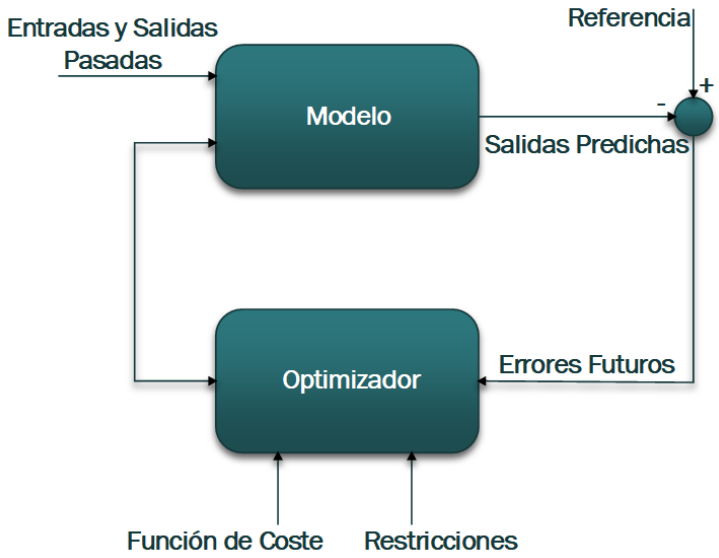


Figura 20. Estructura de los Controladores Predictivos

Fuente: (Limón Marruedo, 2002)

2.2.5.4.1. Modelo de Predicción

Es el modelo matemático que describe el comportamiento del proceso, además es el encargado de proporcionar predicciones en el comportamiento de la dinámica del mismo.

Horizonte de predicción (Hp): Es el encargado de predecir las salidas futuras, usando el modelo del proceso. Los valores de las salidas pronosticadas $y(t+1)$ dependen directamente del estado del proceso en el tiempo (t) actual y de los valores de las señales de entrada y salida pasadas.

Horizonte de control (Hc): se calculan varios movimientos futuros de la señal de control $u(t+1)$, donde $Hc \leq Hp$, véase la Figura 21.

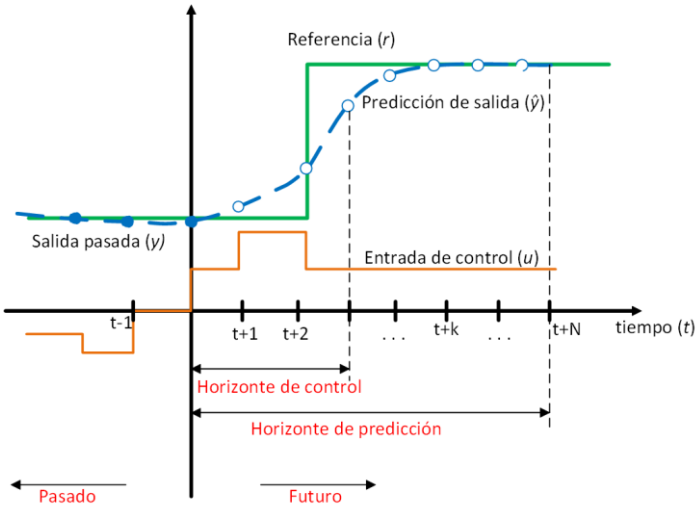


Figura 21. Modelo de predicción del control predictivo basado en el modelo

El control predictivo basado en el modelo posee diferentes estrategias de control, donde se puede utilizar diferentes modelos que representan la relación de la entrada con la salida medible, además de considerar perturbaciones del proceso; entre los modelos más usados están:

- Respuesta Impulsional
- Respuesta ante escalón
- Respuesta de Transferencia
- Espacio de Estados

Para el desarrollo del trabajo se optó por el método de espacio de estados, porque es ideal para controles multivariables y ofrece la facilidad de analizar la estructura interna del proceso (Vaca & Curay, 2015). La representación en espacio de estados para el MPC se representa de la siguiente forma:

$$x(t+1) = Ax(t) + Bu(t) \tag{2.5}$$

$$y(t) = Cx(t) + Du(t) \tag{2.6}$$

El modelo se ve representado por las ecuaciones (2.5) y (2.6)

$$\overbrace{\begin{bmatrix}\Delta x(t+1)\\ y(t)\end{bmatrix}}^{x(t+1)} = \overbrace{\begin{bmatrix}A & o_m^T\\ CA & 1\end{bmatrix}}^{A_e} \overbrace{\begin{bmatrix}\Delta x(t)\\ y(t)\end{bmatrix}}^{x(t)} + \overbrace{\begin{bmatrix}B\\ C.B\end{bmatrix}}^{B_e}\Delta u(t) \tag{2.7}$$

$$y(t) = \overbrace{\begin{bmatrix}O_m & 1\end{bmatrix}}^{C_e}\begin{bmatrix}\Delta x(t)\\ y(t)\end{bmatrix} \tag{2.8}$$

Desarrollando el modelo se calcula las predicciones de estado y salida al instante actual t_i, véase ecuaciones (2.9), (2.10) y (2.11).

$$x(t_i+1|t_i) = A_e x(t_i) + B_e \Delta u(t_i) \tag{2.9}$$

$$x(t_i+2|t_i) = A_e^2 x(t_i) + A_e B_e \Delta u(t_i) + B_e \Delta u(t_i+1) \tag{2.10}$$

$$x(t_i+N|t_i) = A_e^N x(t_i) + A_e^{N-1} B_e \Delta u(t_i) + A_e^{N-M} B_e \Delta u(t_i+M-1) \tag{2.11}$$

N representa el horizonte de predicción y M es el horizonte de control. Para las predicciones de salida se usa el mismo criterio de las ecuaciones anteriores, véase ecuaciones (2.12), (2.13) y (2.14).

$$y(t_i+1|t_i) = C_e A_e x(t_i) + C_e B_e \Delta u(t_i) \tag{2.12}$$

$$y(t_i+2|t_i) = C_e A_e^2 x(t_i) + C_e A_e B_e \Delta u(t_i) + B_e \Delta u(t_i+1) \tag{2.13}$$

$$y(t_i+N|t_i) = C_e A_e^N x(t_i) + C_e A_e^{N-1} B_e \Delta u(t_i) + C_e A_e^{N-2} B_e \Delta u(t_i+1) + C_e A_e^{N-M} B_e \Delta u(t_i+M-1) \tag{2.14}$$

El conjunto de predicciones está representado de la siguiente forma:

$$Y = Fx(t_i) + \Phi \Delta U \tag{2.15}$$

F y Φ están basadas en las matrices del modelo aumentado, véase las ecuaciones (2.16) y (2.17) (Aguilar & Ortiz, 2017).

$$F = \begin{bmatrix} C_e A_e \\ C_e A_e^2 \\ C_e A_e^3 \\ \vdots \\ C_e A_e^N \end{bmatrix} \tag{2.16}$$

$$F = \begin{bmatrix} C_e B_e & 0 & \dots & 0 \\ C_e A_e B_e & C_e B_e & \dots & 0 \\ C_e A_e^2 B_e & C_e A_e B_e & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ C_e A_e^{N-1} B_e & C_e A_e^{N-2} B_e & \dots & C_e A_e^{N-M} B_e \end{bmatrix} \tag{2.17}$$

2.2.5.4.2. Función objetivo o Función de coste

Esta función indica el criterio a optimizar. Es una función definida positiva que expresa el coste asociado a la evolución del sistema a través del horizonte de predicción. Este valor trata de exponer el grado de desempeño de las especificaciones estáticas y dinámicas del proceso, se representa por la ecuación (2.18) (Blasco Ferragud, s.f.):

$$J(y,u) = (R_s - Y)^T (R_s - Y) + \Delta U^T R \Delta U \tag{2.18}$$

Donde R_s, es el punto de ajuste y ΔU, son parámetros de la señal.

2.2.5.4.3. Ley de Control

Esta ley de control se obtiene a partir de un optimizador, para ofrecer una mejora en la función objetivo. El problema de la optimización tiene como variables de decisión las acciones que se producen a lo largo del horizonte de control; una vez que se obtiene la solución, según la estrategia del horizonte deslizante se considera el primer elemento del vector ΔU, mientras que los demás son omitidos; con este elemento ya se puede calcular una acción de control válida para la planta en cada instante de muestreo (t_i), véase ecuación (2.19).

$$\Delta U = H \Phi^T \Phi [R_s . r(t_i) - F x(t_i)] \tag{2.19}$$

2.2.5.5. Elementos de la Optimización

- **Criterio a optimizar:** Expresión matemática que muestra el valor cuantitativo del funcionamiento del sistema a optimizar.
- **Variables de decisión:** Representan las decisiones que se toman para afectar el valor de la función objetivo.
- **Restricciones:** Son el conjunto de relaciones que indican los límites físicos o de seguridad donde funciona el actuador del sistema. Estas medidas evitan aumentar gastos económicos.

2.2.5.6. Principio del Horizonte Deslizante

Implica que en cada instante de tiempo t se determina un problema de control óptimo sobre un horizonte finito futuro N, donde una función f que pondera la diferencia entre la salida, la referencia y el esfuerzo de control, véase la ecuación (2.20):

$$\min_{u} f(|y - r|, |u|) \tag{2.20}$$

Donde y es la salida, r es la referencia y u es el esfuerzo de control, las cuales están sujetas a restricciones:

$$\begin{aligned} y_{min} \leq y \leq y_{max} \\ u_{min} \leq u \leq u_{max} \\ \Delta u_{min} \leq u \leq \Delta u_{max} \end{aligned} \tag{2.21}$$

En la ecuación (2.22) se observa que cada instante k, a partir del modelo se calcula una secuencia de control, donde solo el primer valor $u(k|k)$ es implementado:

$$u(k|k), \qquad u(k+1|k), \qquad u(k+2|k), \dots, u(k+N|k) \tag{2.22}$$

Luego de que la planta obtenga nuevas medidas se repite el proceso de optimización (Feroldi, 2012).

CAPÍTULO III

3. DISEÑO Y SIMULACIÓN DE LOS CONTROLADORES

3.1. Introducción

Con la finalidad de realizar el diseño de los controlares avanzados propuestos, en este capítulo se presenta el procedimiento adecuado para la obtención del modelo matemático del proceso. Para ello es necesario conocer detalladamente los componentes que forman el sistema modular de la estación de flujo de caudal, sus principales características, las unidades y rangos en los que operan tanto la entrada como la salida, su comportamiento y posibles perturbaciones, para poder elaborar los algoritmos de control dependiendo del requerimiento que tenga el usuario y posteriormente realizar las pruebas experimentales correspondientes.

3.2. Descripción de la estación de flujo de caudal

La planta de flujo de caudal tiene un tanque cilíndrico de metal con una capacidad de 25 galones, tubería de acero galvanizado de ¾ (ya que el transmisor lo requiere), una válvula manual de paso que permite el paso de fluido del tanque a una bomba centrífuga THEBE de ½ Hp.

Además, cuenta con un transmisor magnético Rosemount 8732E que mide la variable flujo y a su salida produce una señal estándar de 4 a 20 mA, la cual es interpretada por un PLC Siemens S7-1500. Dicho controlador emite una señal de voltaje entre 0 y 10 V que se envía al variador de frecuencia Delta VFD004E23A y así manipular el flujo de caudal que circula a través del proceso.

Para corroborar el correcto funcionamiento del algoritmo de control implementado es necesario su visualización, para ello esta planta posee un display propio del transmisor antes mencionado. También posee un rotámetro graduado, cuyo émbolo muestra el flujo de caudal entre 0 y 40 litros por minuto (LPM). Adicional a esto, el PLC está conectado a una pantalla táctil donde se encuentra elaborado un HMI que le permite al usuario interactuar con el proceso. A continuación, en la Figura 22 se muestra el diagrama de bloques de la estación (Carvajal & Proaño, 2015).

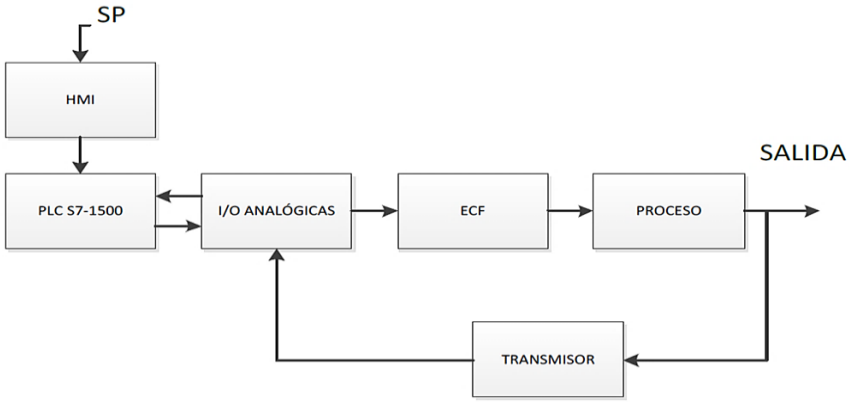


Figura 22. Diagrama de Bloques de la Estación

Fuente: (Carvajal & Proaño, 2015)

Cabe aclara que a pesar que la planta cuenta con un PLC propio y un TOUCH SCREEN, un eje prioritario de este trabajo es implementar algoritmos de control utilizando tarjetas embebidas de bajo costo y un HMI en una computadora personal utilizando licencias estudiantiles, de esta forma se pretende brindar la misma solución de ingeniería con una inversión menor.

3.3. Diseño del algoritmo de control borroso

Para que los resultados sean óptimos, es necesario que el diseño de los conjuntos borrosos y de la base de reglas de inferencia sea el adecuado. Un mayor número de reglas y cálculos profundos producirían un mayor consumo de recursos y una mayor latencia, por lo cual se trató de realizar un diseño eficiente procurando reducir el grado de dificultad para una mejor comprensión.

3.3.1. Identificación de las variables de entrada y salida

Para esta aplicación se desea que el valor de la variable de proceso (PV) trate de igualar al valor de la variable deseada (SP). Éstas son las variables de entrada y como variable de salida un valor de voltaje, dando como resultado un sistema MISO (múltiples entradas - una salida). Para facilitar el proceso en lugar de utilizar dos entradas, se trabaja con la señal de error producto de la diferencia entre el SP y el PV, con ello se ha simplificado el sistema a uno del tipo SISO (una entrada – una salida).

$$Error = Flujo\ de\ caudal\ deseado\ (SP) - Flujo\ de\ caudal\ medido\ (PV)$$

Se ha determinado que dentro de los algoritmos de control las variables PV y SP tengan un rango de operación entre 0 y 50. Para definir el universo de discurso se han evaluado las máximas condiciones que se pueden suscitar, dando como resultado valores entre **-150 y 150**, como se muestra a continuación:

$$Error = SP_{MÁXIMO} - PV_{MÍNIMO}$$
$$Error = 150 - 0$$
$$Error = 150$$

$$Error = SP_{MÍNIMO} - PV_{MÁXIMO}$$
$$Error = 0 - 150$$
$$Error = -150$$

3.3.2. Emborronado

Una vez que se han definido las variables de entrada y salida se procede a emborronarlas (proceso más conocido como fusificación) para que el controlador proceda a procesarlas.

La variable lingüística de entrada será el error denominada "err_{caudal}" y para los conjuntos borrosos se han definido 7 funciones de pertenencia del tipo trapezoidal y triangular, como se presenta en la Tabla 7 y en la Figura 23.

Tabla 7.
Conjuntos borrosos de entrada

Nombre del conjunto difuso	Descripción	Intervalo
NA	Error negativo alto	-150 a -75
NM	Error negativo medio	-112.5 a -37.5
NB	Error negativo bajo	-75 a 0
CE	Error neutro	-37.5 a 37.5
PB	Error positivo bajo	0 a 75
PM	Error positivo medio	37.5 112.5
PA	Error positivo alto	75 a 150

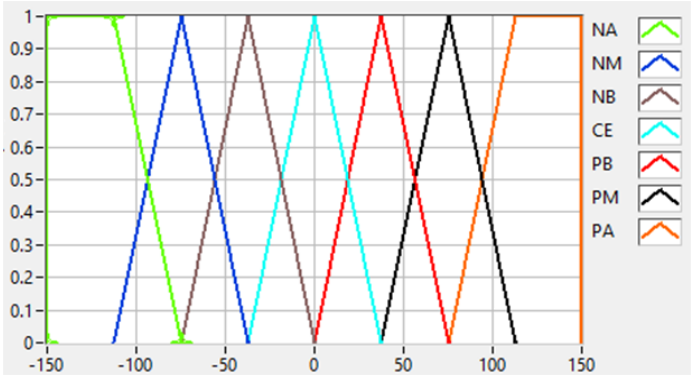


Figura 23. Conjuntos borrosos de entrada de la variable lingüística err_{caudal}

La variable lingüística de salida será el voltaje que va al variador denominada “Voltaje” y el universo de discurso ha sido definido en un intervalo de -20 a 10. Para los conjuntos borrosos se han definido 7 funciones de pertenencia del tipo trapezoidal y triangular, como se presenta en la Figura 24 y en la Tabla 8 respectivamente:

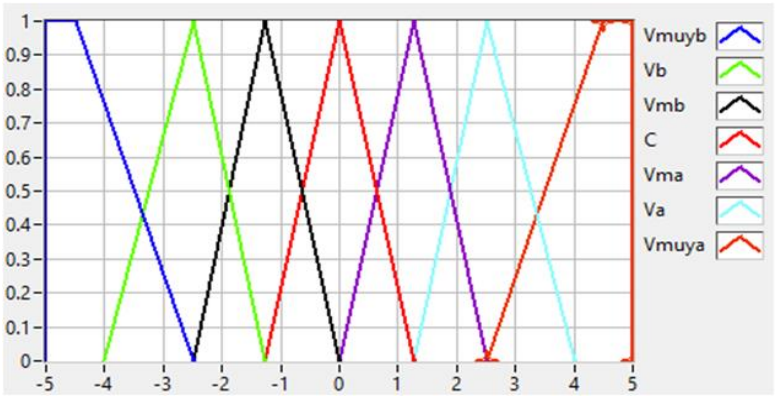


Figura 24. Conjuntos borrosos de salida de la variable lingüística Voltaje

Tabla 8.
Conjuntos borrosos de salida

Nombre del conjunto difuso	Descripción	Intervalo
Vmuyb	Voltaje muy bajo	-5 a -2.5
Vb	Voltaje bajo	-4 a -1.275
Vmb	Voltaje medio bajo	-2.5 a 0
C	Voltaje cero	-1.275 a 1.275
Vma	Voltaje medio alto	0 a 2.5
Va	Voltaje alto	1.275 a 4
T	Voltaje terminal	2.5 a 5

3.3.3. Base de reglas

La elaboración del conjunto de reglas borrosas se basa en el conocimiento "general" de la variación del flujo de caudal en la estación. En base a esto asociar las combinaciones probables de los conjuntos de entrada a un valor de salida. Se ha elegido diseñar un controlador con inferencia de tipo Mamdani, donde las reglas quedaron de la siguiente forma:

1. $\boldsymbol{SI}\ err_{caudal}\ es\ NA, \boldsymbol{ENTONCES}\ el\ voltaje\ del\ variador\ es\ Vmuyb$
2. $\boldsymbol{SI}\ err_{caudal}\ es\ NM, \boldsymbol{ENTONCES}\ el\ voltaje\ del\ variador\ es\ Vb$
3. $\boldsymbol{SI}\ err_{caudal}\ es\ NB, \boldsymbol{ENTONCES}\ el\ voltaje\ del\ variador\ es\ Vmb$
4. $\boldsymbol{SI}\ err_{caudal}\ es\ CE, \boldsymbol{ENTONCES}\ el\ voltaje\ del\ variador\ es\ C$
5. $\boldsymbol{SI}\ err_{caudal}\ es\ PB, \boldsymbol{ENTONCES}\ el\ voltaje\ del\ variador\ es\ Vma$
6. $\boldsymbol{SI}\ err_{caudal}\ es\ PM, \boldsymbol{ENTONCES}\ el\ voltaje\ del\ variador\ es\ Va$
7. $\boldsymbol{SI}\ err_{caudal}\ es\ PA, \boldsymbol{ENTONCES}\ el\ voltaje\ del\ variador\ es\ T$

El mismo conjunto de reglas se presenta en forma tabular en la Tabla 9, donde se exponen a lo largo de los ejes el error (eje vertical) y el error acumulado (eje horizontal) y los consecuentes están dentro de la tabla. De esta forma las simetrías se encuentran con mayor facilidad y si se encuentra una celda vacía se determina rápidamente la ausencia de una regla y así evitar inconvenientes. En cuanto al error acumulado N significa negativo, O

cero y P positivo. Para cada regla, el motor de inferencia encuentra el valor de pertenencia donde la línea vertical interseca una función de pertenencia (Cisneros, 2016).

Tabla 9.
Conjunto de reglas expresado en forma tabular

	N	O	P
NA	T	T	T
NM	Va	Va	Vma
NB	Vma	Vma	Vma
CE	Vma	C	Vmb
PB	Vmb	Vmb	Vb
PM	Vb	Vb	Vb
PA	Vb	Vmuyb	Vmuyb

3.3.4. Desemborronado

El conjunto borroso resultante debe pasar a ser un número único que represente una señal de control que será introducida en la planta a controlar. Para ello se define el grado de pertenencia (entre 0 y 1) que posee un valor de entrada en los conjuntos difusos de entrada correspondientes y se lo proyecta en los conjuntos borrosos de salida, se forma un área plana. Para calcular el valor numérico correspondiente se ha realizado el cálculo de centro de área o gravedad como método de desemborronado, el mismo que se calcula mediante la siguiente ecuación (3.1):

$$\mu_{COG} = \frac{\sum_i \mu_c X_i(x_i)}{\sum_i \mu_c(x_i)} \qquad (3.1)$$

3.4. Diagrama de bloques y flujo del sistema

3.4.1. Diagrama de bloques de la comunicación OPC entre Arduino y LabVIEW

En la Figura 25 se detalla el diagrama de bloques, que explica la estructura de comunicación entre Arduino y el HMI desarrollado en LabVIEW.

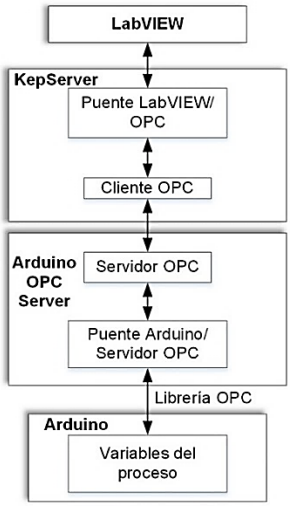


Figura 25. Diagrama de bloques de la comunicación Arduino/LabVIEW

3.4.2. Diagrama de flujo del proceso de control

La Figura 26 muestra de manera detallada el proceso de control, empezando con el tratamiento de la señal hasta la aplicación del control.

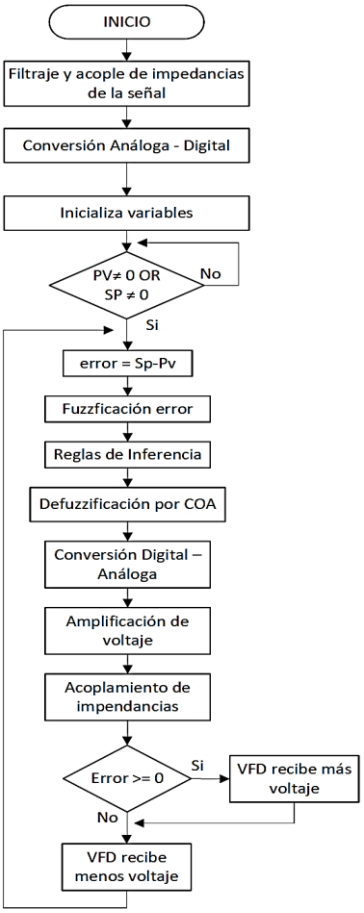


Figura 26. Diagrama de flujo del proceso de control de flujo de caudal

3.4.3. Diagrama de flujo entre las tarjetas embebidas y LabVIEW

En la Figura 27 se describe el flujo de comunicación entre las tarjetas embebidas y LabVIEW para analizar y comparar su desempeño con el controlador borroso implementado.

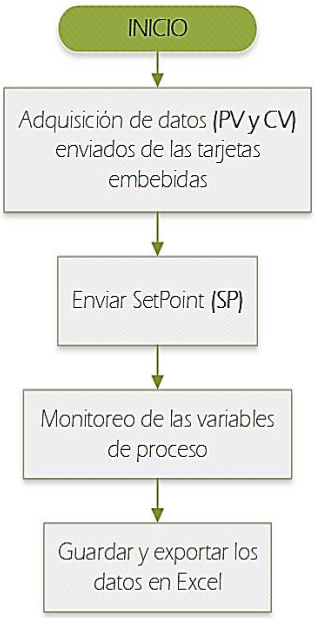


Figura 27. Diagrama de flujo en LabVIEW entre las tarjetas embebidas y el software

3.5. Comunicación OPC

Arduino ofrece la posibilidad de establecer una comunicación con el HMI creado en LabVIEW mediante un servidor OPC; para su configuración previamente se necesita descargar la librería OPC, mediante el **"Gestor de Librerías"** del IDE de Arduino y el programa gratuito **"Arduino OPC Server"**, el cual está disponible en el siguiente enlace: https://www.st4makers.com/download-opc-server-for-arduino. El servidor OPC de Arduino sirve de puerta de enlace entre el IDE de Arduino y el software KepServer Ex 5 que hace las veces de cliente OPC, para el software LabVIEW.

3.5.1. Instalación y configuración de Arduino OPC Server 1.9

Antes de iniciar con la instalación del servidor OPC de Arduino es indispensable instalar los componentes principales de OPC, los cuales están disponibles en el siguiente enlace: https://opcfoundation.org/developer-tools/developer-kits-classic/core-components; para acceder a la descarga es necesario crear un usuario y contraseña con fines de verificación de la página.

1. Descomprima y ejecute el archivo ArduinoOPCServer.exe. Luego de ejecutarse el archivo generará dos archivos nuevos (ver Figura 28).

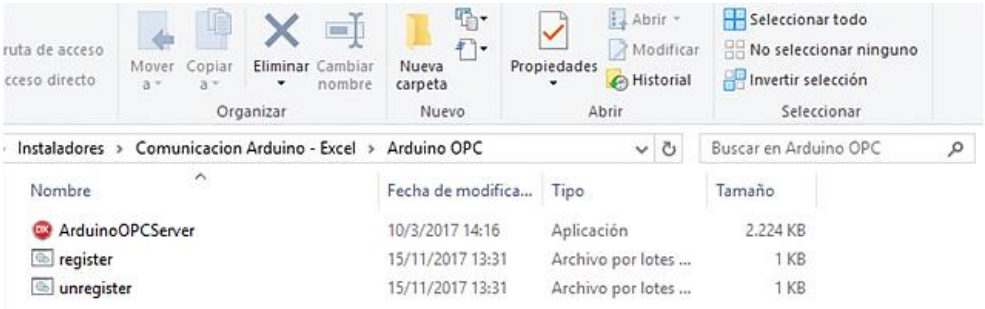


Figura 28. Archivos creados al ejecutar el software ArduinoOPCServer

2. Los archivos que se crean sirven para registrar el servidor en el sistema operativo, luego configurar el servidor dirigiéndose a la pestaña Configuration. Ahí seleccionar el puerto serial, la velocidad de comunicación y el intervalo de lectura como se muestra en la Figura 29.

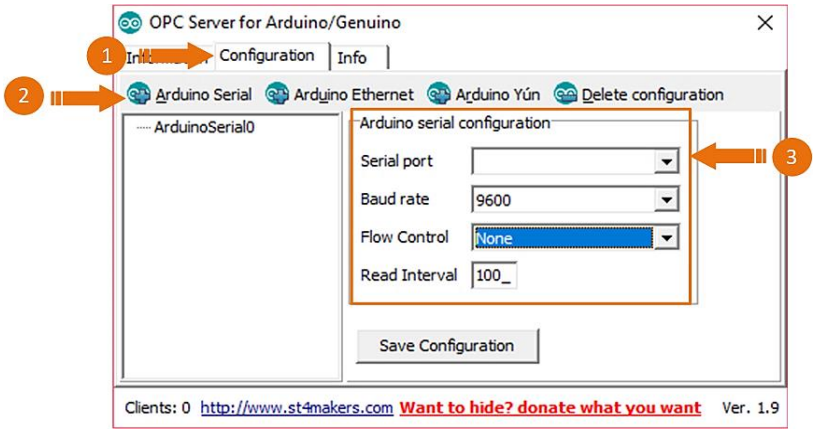


Figura 29. Configuración del servidor OPC de Arduino

3. Salvar la configuración y ejecutar como administrador el archivo **“register”,** para registrar en el sistema operativo los cambios que se realizaron en la comunicación. Estos pasos se realizan una sola vez o cuando los parámetros de comunicación sean cambiados, como se explica en la Figura 30.

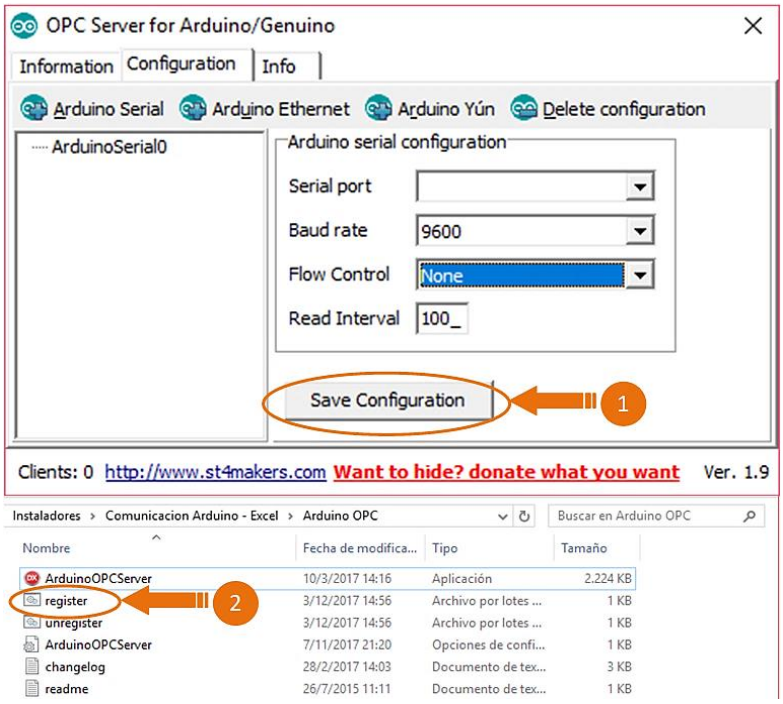


Figura 30. Salvar y registrar configuración de comunicación en el sistema operativo

3.5.2. Añadir la librería de OPC en el software de Arduino

1. Ejecutar el IDE de Arduino y seleccione la pestaña “Programa”, en esta opción elija “Incluir Librería” y luego “Gestionar Librerías”.

2. En la ventana que se despliega dirigirse a la barra de búsqueda y escribir OPC para localizar la librería necesaria; seleccionar la última versión y darle a instalar (ver Figura 31).

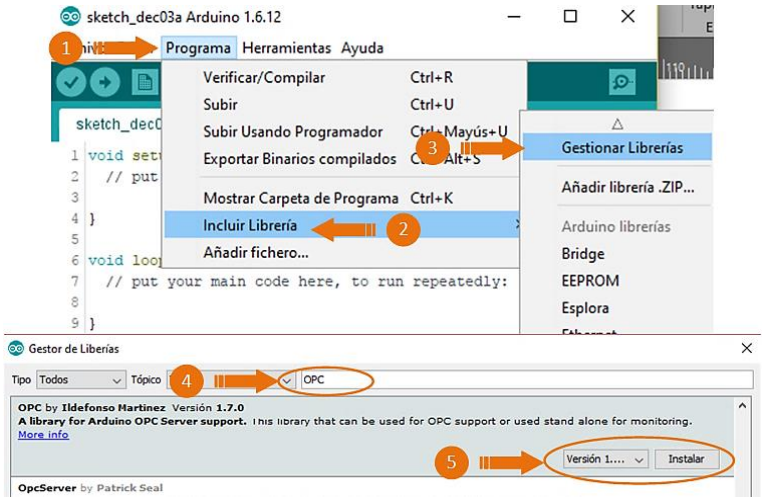


Figura 31. Instalación de la librería OPC para Arduino

3.5.3. Configuración del cliente OPC en el software KEPServerEx 5

En el software KEPServerEx 5 se configura el canal y dispositivo que permita enviar los valores de las variables de proceso del control realizado en Arduino y así visualizarlas en LabVIEW.

a. Creación del Canal del cliente OPC

1. Crear un nuevo canal y darle un nombre para identificarlo fácilmente (ver Figura 32).

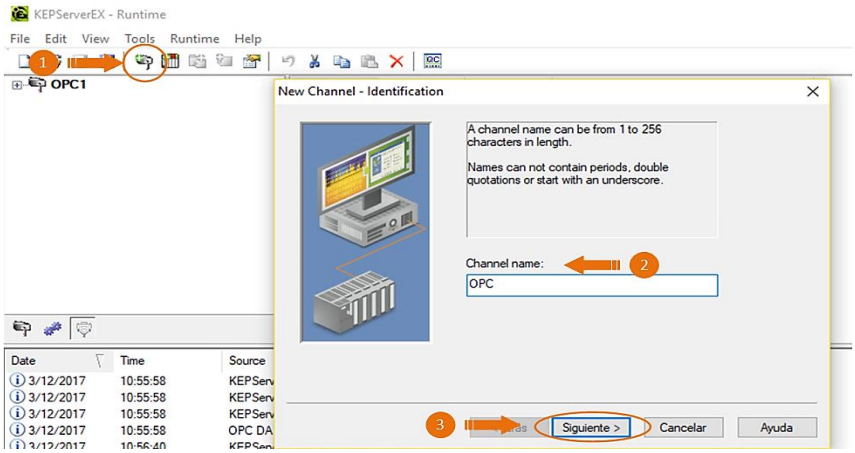


Figura 32. Creación de un nuevo canal en el cliente OPC

2. Seleccionar el controlador del dispositivo; para este proyecto el controlador a elegir es OPC DA Client, dar clic en siguiente hasta llegar a la ventana de selección del servidor OPC (ver Figura 33).

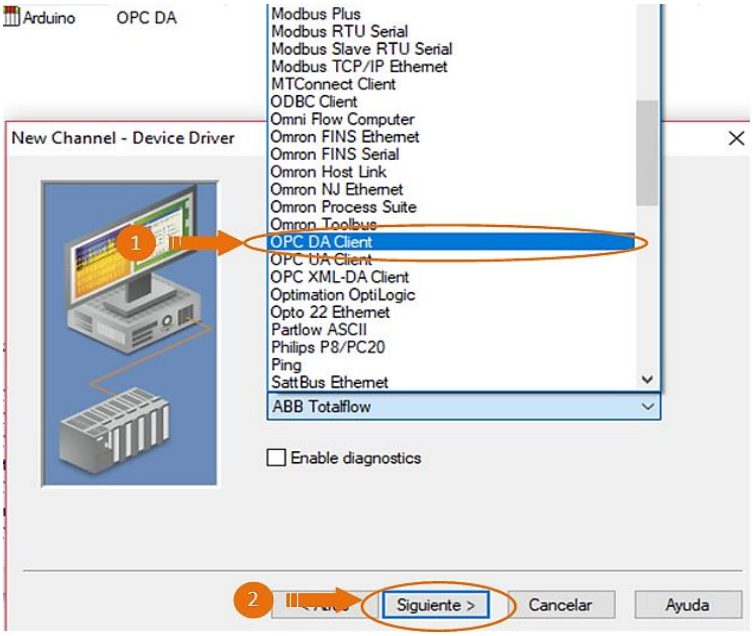


Figura 33. Configuración del controlador del dispositivo

3. En la ventana del wizard de ayuda seleccionar el servidor OPC con el cual va a conectarse, como se trata de una comunicación local se despliega la pestaña Local Machine y escoger ArduinoOPCServer.1 (ver Figura 34).

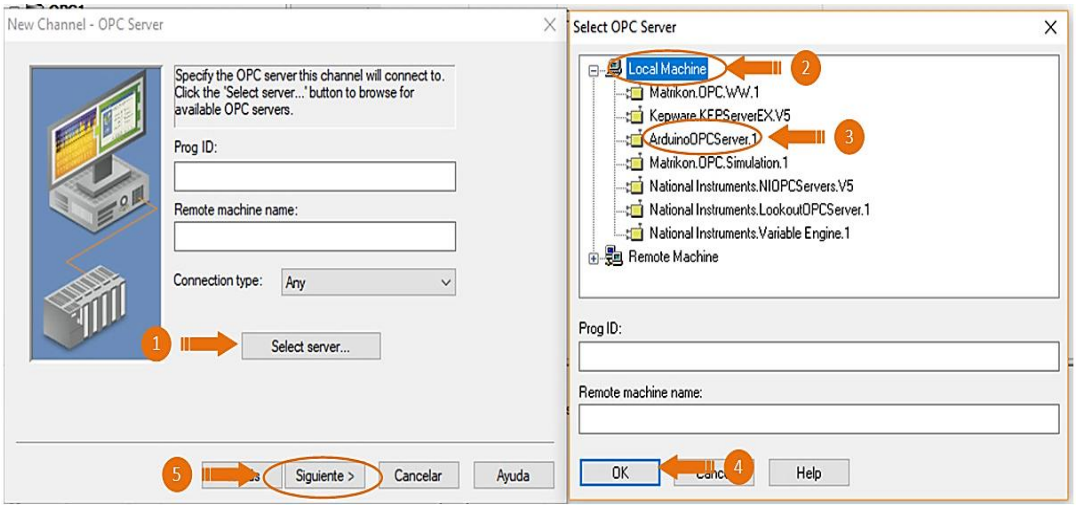


Figura 34. Selección del servidor OPC

4. Avanzar en las ventanas de ayuda del wizard (no es necesario modificar los parámetros) hasta finalizar la creación del canal.

b. Creación del Dispositivo OPC y los tags de comunicación

1. Crear un nuevo dispositivo en el canal que se creó anteriormente (ver Figura 35).

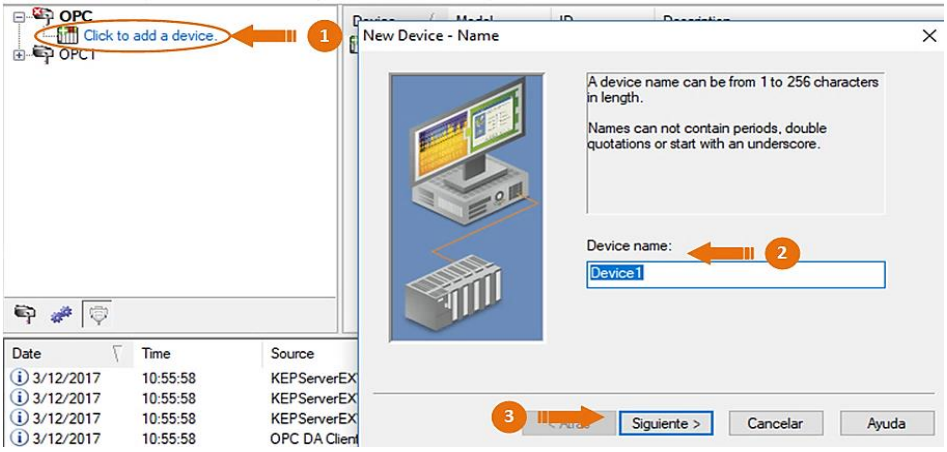


Figura 35. Creación de un nuevo dispositivo

2. Clic en siguiente hasta llegar a la ventana de importar ítems, cabe mencionar que en las ventanas anteriores no es necesario modificar ningún parámetro para obtener una buena comunicación.

3. Importar los ítems que deben estar previamente compilados y cargados a la tarjeta Arduino para ser reconocidos por el software; añadir uno a uno los tags que representan a las variables de proceso (ver Figura 36).

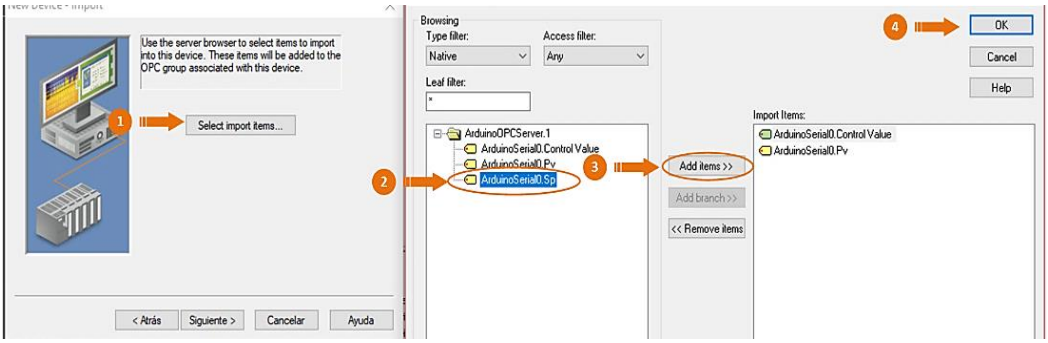


Figura 36. Importación de ítems para crear los tags

4. Verificar que la calidad de comunicación sea buena (ver Figura 37).

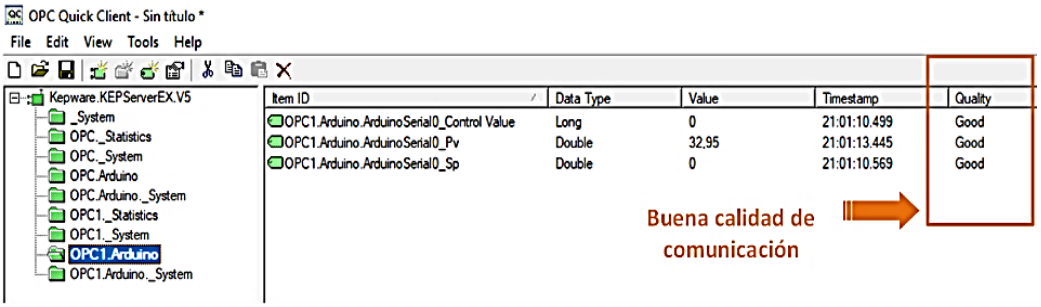


Figura 37. Verificación de la calidad de comunicación

3.6. Diseño del algoritmo del Control Predictivo por Modelo (MPC)

3.6.1. Modelamiento de la Estación de Caudal

Para diseñar el MPC (Control Predictivo basado en el Modelo), como se mencionó en el apartado de fundamentación teórica es necesario obtener un modelo matemático óptimo, por lo tanto, es fundamental obtener datos con la información suficiente que describa el modelo de la planta.

Por esta razón es necesario definir los rangos de trabajo tanto de la entrada y la salida, para luego con la ayuda del toolbox de identificación de sistemas (System Identification) de MATLAB obtener un modelo de la planta representado en función de espacio de estado, por ser el método más fácil para trabajar.

3.6.2. Definición de rangos del sistema

Primero se debe analizar los rangos en los que la planta trabaja, es decir los niveles de flujo donde las señales del transmisor marquen 4 y 20 mA, valores que serán convertidos a voltaje mediante una resistencia en serie y que en el capítulo siguiente se explicará más a fondo.

Hardware

La planta como se mencionó en el párrafo anterior está limitada por el rango de 4 – 20 mA. La entrada trabaja en un rango de 0 a 10 V, valores que definen la acción del actuador (0 – 100%), mientras tanto la salida con la que está configurada el transmisor es de 10 a 40 LPM.

Software

Para la adquisición de datos se usa LabVIEW, mientras que para la obtención del modelo matemático se usa MATLAB, programas en los que no existe ninguna limitación en su manejo.

3.6.3. Identificación del Sistema

Mediante un programa cargado en la tarjeta Arduino UNO R3 y con la ayuda del software LabVIEW, para la visualización y guardado de datos se pudo adquirir los datos de entrada y salida, luego con el comando "systemIdentification" en MATLAB se logra obtener el modelo de la planta en función de espacio de estados, para eso es necesario seguir los siguientes pasos.

1. Con los datos obtenidos se debe exportarlos hacia el workspace de MATLAB, para eso se debe crear dos nuevas variables para la entrada (SP) y la salida (PV) respectivamente.

2. Para una mayor comodidad es recomendable renombrar las variables creadas y luego copiar los datos de entrada y salida obtenidos anteriormente.

3. Ejecutar el comando "systemIdentification" en la ventana "Command Window" de MATLAB, se desplegará una ventana (ver Figura 38).

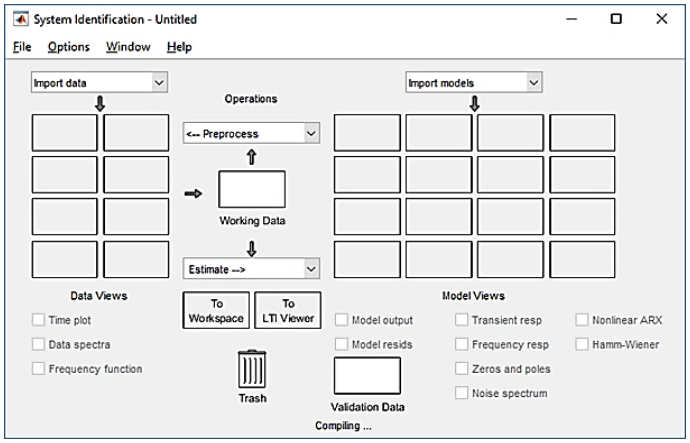


Figura 38. Ventana del toolbox para la identificación de sistemas

4. Cargar los datos guardados previamente, para eso dirigirse a “Import data” y seleccionar “Time domain data”, en la nueva ventana que se despliega llenar los campos que se requiere, y después presionar el botón “Import” (ver Figura 39).

- **Input:** Variable de entrada (SP)
- **Output:** Variable de salida (PV)
- **Starting time:** Tiempo de inicio de la toma de muestras (0)
- **Sample time:** Tiempo de muestreo de la entrada y salida (0.108)

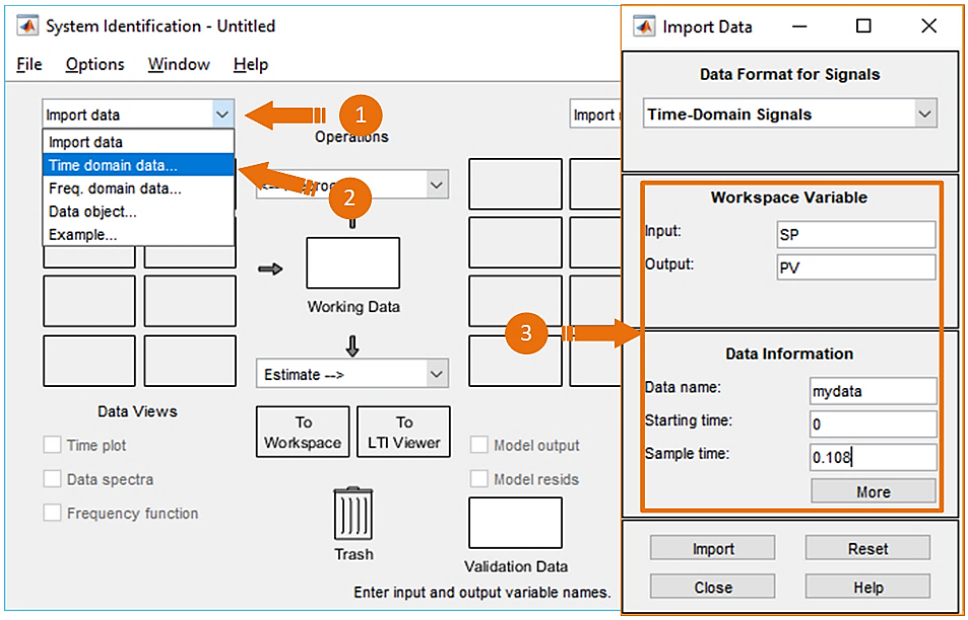


Figura 39. Importación de los datos a la función systemIdentification

5. Con los datos exportados se puede estimar un modelo matemático con el mayor porcentaje de similitud; como se mencionó anteriormente los modelos se estimarán en ecuaciones de estado debido a que el toolbox para diseñar el MPC en MATLAB trabaja con este tipo de ecuaciones.

Para estimar el mejor modelo se tomó 4 muestras diferentes y para cada una se determinó 5 modelos diferentes, como se observa en la Figura 40.

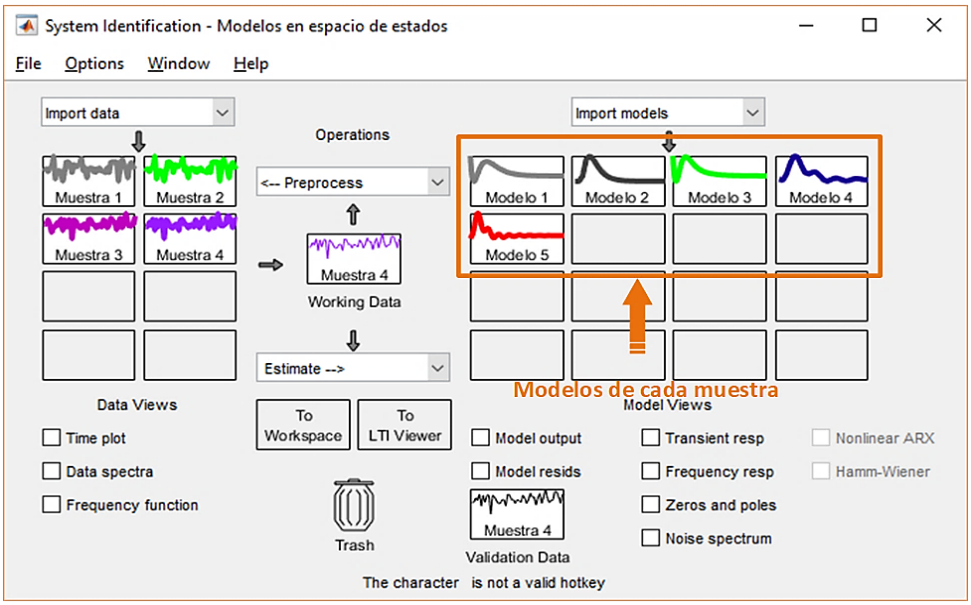


Figura 40. Modelos matemáticos obtenidos con el toolbox de identificación de sistemas

En la Tabla 10 se muestra un resumen de los porcentajes obtenidos en cada modelo, además de señalar cuál de ellos obtuvo el mayor porcentaje de similitud con la señal original. Para diseñar el MPC se eligió el modelo 3 de la muestra número 1 ya que es la que posee el porcentaje más alto de similitud; se exporta el modelo al workspace de MATLAB.

Tabla 10.
Tabla de comparación de modelos

N.º Muestra	Modelos	Porcentaje de similitud (%)
Muestra 1	Modelo 1	94,63
	Modelo 2	94,63
	Modelo 3	**94,94**
	Modelo 4	94,65
	Modelo 5	94,65
Muestra 2	Modelo 1	90,6
	Modelo 2	86,93
	Modelo 3	81,85

Continúa

	Modelo 4	86,96
	Modelo 5	88,88
Muestra 3	Modelo 1	81,72
	Modelo 2	86,68
	Modelo 3	90,15
	Modelo 4	82,61
	Modelo 5	93,6
Muestra 4	Modelo 1	56,9
	Modelo 2	53,77
	Modelo 3	74,58
	Modelo 4	91,06
	Modelo 5	93,81
Muestra Seleccionada		94,94

La ecuación en espacio de estados queda representada de la siguiente forma:

Tiempo de muestreo = Ts = 0.108.

$$x(t+Ts) = Ax(t) + Bu(t) + Ke(t) \tag{3.2}$$

$$y(t) = Cx(t) + Du(t) \tag{3.3}$$

$$\begin{bmatrix} \dot{x_1}(t) \\ \dot{x_2}(t) \\ \dot{x_3}(t) \\ \dot{x_4}(t) \end{bmatrix} = \begin{bmatrix} -1.3118 & 0.8394 & 0.1933 & -2.0197 \\ -2.1835 & -0.1438 & -2.1793 & 9.7886 \\ -0.0540 & 3.2032 & 1.0284 & 11.6407 \\ -5.3352 & -9.0445 & -7.4187 & -19.3374 \end{bmatrix} \begin{bmatrix} x_1(t) \\ x_2(t) \\ x_3(t) \\ x_4(t) \end{bmatrix} + \begin{bmatrix} 0.0292 \\ -0.0326 \\ -0.2020 \\ 0.5903 \end{bmatrix} \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ u_4(t) \end{bmatrix} + \begin{bmatrix} 0.1326 \\ 0.1842 \\ -0.2399 \\ 0.0331 \end{bmatrix} \begin{bmatrix} e_1(t) \\ e_2(t) \\ e_3(t) \\ e_4(t) \end{bmatrix} \tag{3.4}$$

$$\boldsymbol{y_1(t)} = [-53.7225 \quad 0.4858 \quad -1.8423 \quad 0.4425][x_1(t)] + [0][u_1(t)] \tag{3.5}$$

3.6.4. Modelo del MPC para la estación de caudal

Para la simulación del controlador se usa el toolbox MPC Designer de MATLAB, que facilita el desarrollo de dicho controlador, para eso se crea un diagrama de bloques para simular la planta en Simulink. En la Figura 41 se muestra el bloque de MPC y el bloque que simula la planta.

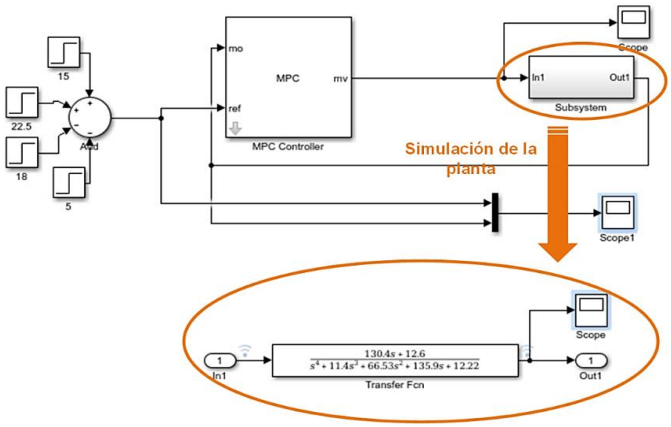


Figura 41. Diagrama de bloques para simular el control MPC

1. En el diseño se accede al bloque MPC y se da clic sobre “Design”, en la nueva ventana que se despliega elegimos “MPC Structure”, para definir los canales del controlador (véase la Figura 42).

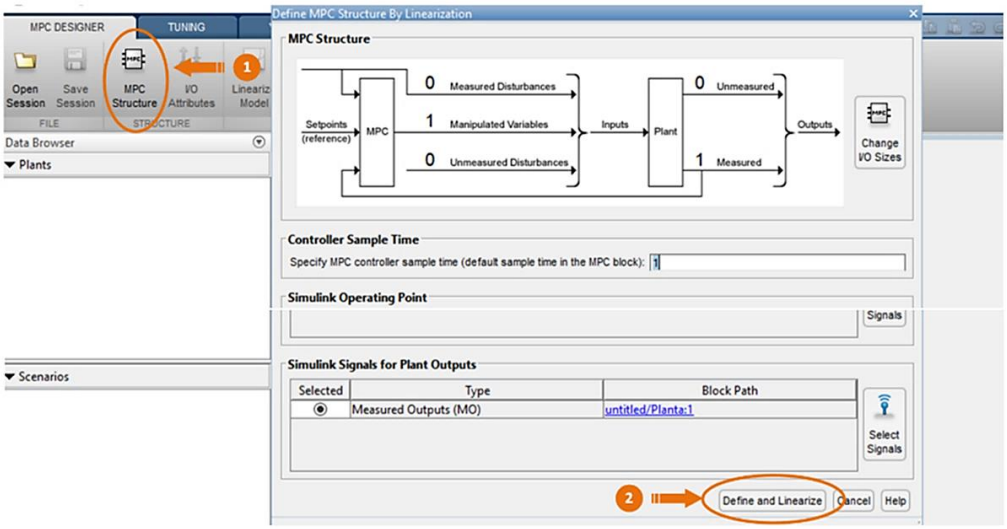


Figura 42. Definición de la estructura del MPC

Antes de importar la planta, la cual fue exportada al workspace desde el toolbox de identificación de sistemas, es necesario crear la ecuación con las matrices de estado; para eso se guarda en variables independientes las matrices de estado obtenidas del modelo exportado, luego se crea la ecuación de estado mediante la función “ss”, y con la función “c2d”, la cual se transforma a una ecuación discreta, véase la Figura 43.

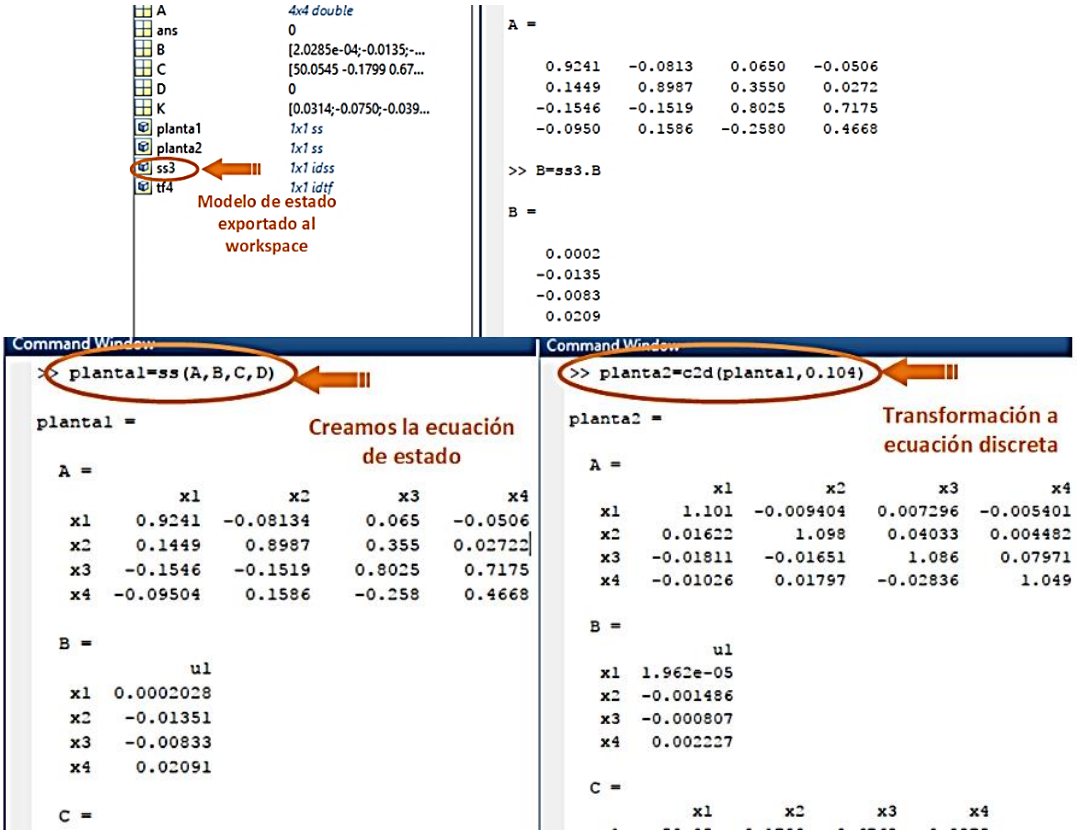


Figura 43. Creación de la ecuación de estado para importar al modelo MPC

2. Importar la planta creada anteriormente y editar el escenario con los valores que trabaja la estación, para obtener una simulación correcta, véase la Figura 44.

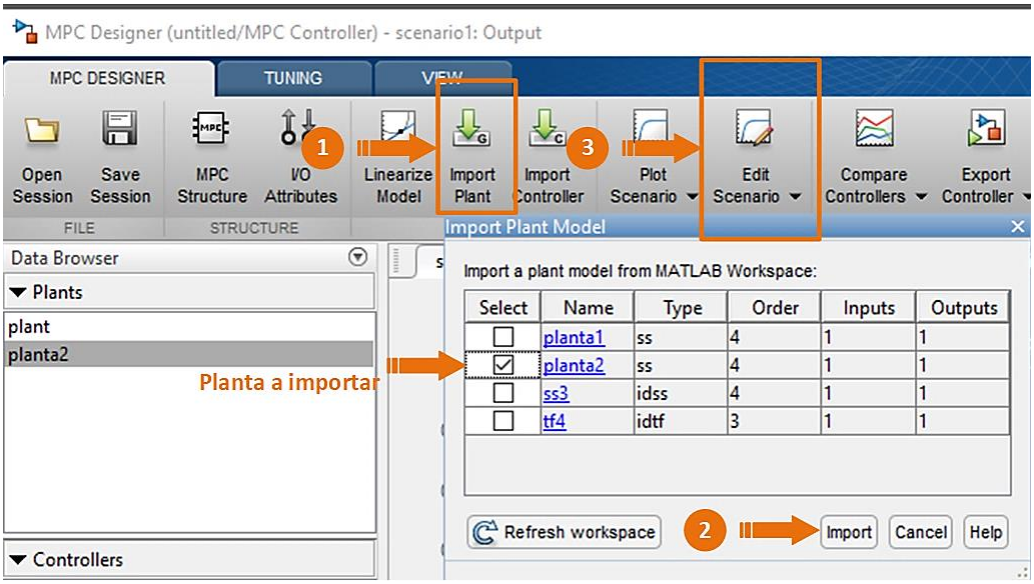


Figura 44. Importación de la planta para la simulación

3. Como se muestra en la Figura 45, al seleccionar la pestaña “TUNING”, se elige la planta que se importa, se modifica el horizonte de predicción y el horizonte de control hasta obtener una respuesta adecuado del controlador.

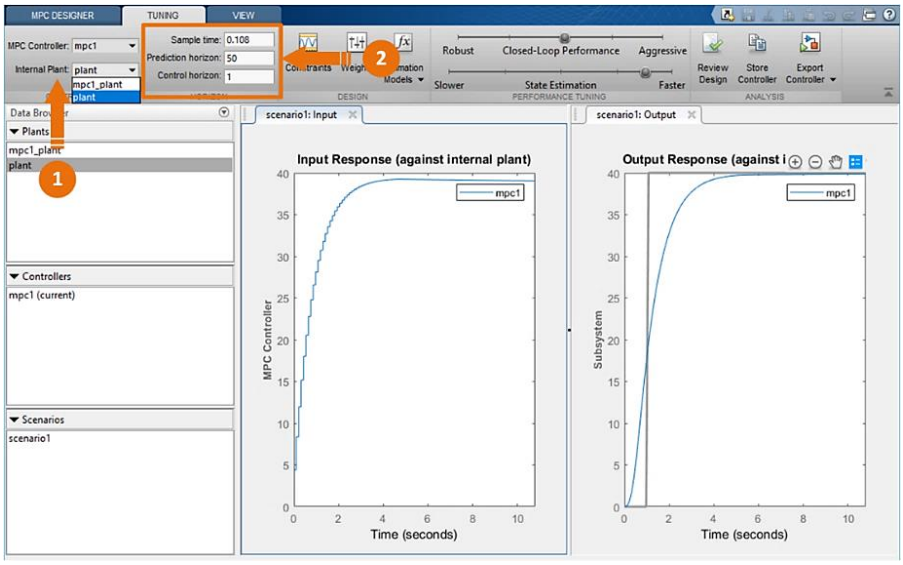


Figura 45. Selección del horizonte de predicción y control adecuado para el modelo

En párrafos anteriores se mencionó que la planta tiene límites de trabajo (10 – 40 LPM), por esta razón se procede a colocar restricciones a la salida **(y)** del modelo, como se puede observar en la Figura 46. Con esta medida se asegura que el controlador no supere los rangos de trabajo de la estación.

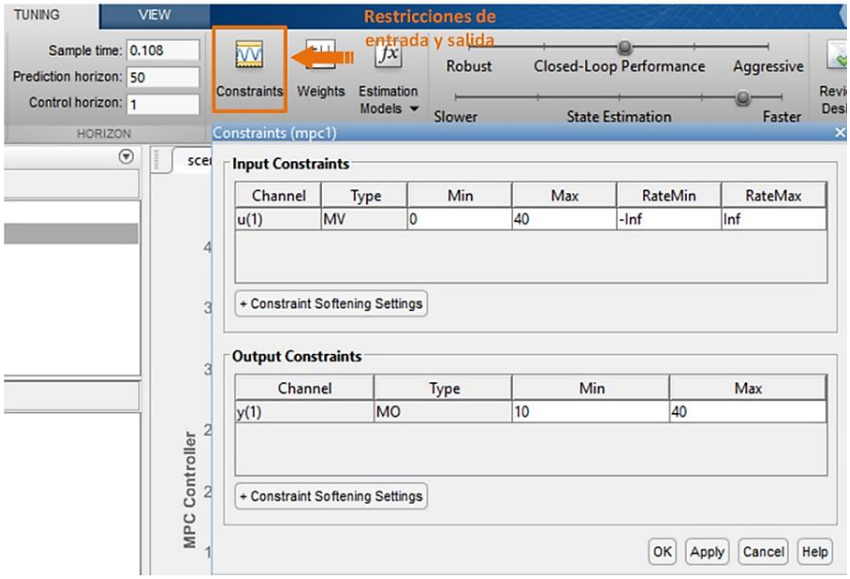


Figura 46. Restricciones colocadas en el modelo

La opción de “Weights”, permite manejar los pesos de entrada y salida, un peso de entrada alto provoca que la variable manipulada **(MV)** se aleje del valor de referencia y cause errores de estado estacionario.

Nota: Modificar el “Rate Weight”, significa establecer una penalidad en el cambio de la MV, en otras palabras, un rate weight alto significa que el controlador provocará cambios más lentos en la variable manipulada. Un peso demasiado alto en la salida también provocará que la misma se aleje del valor de referencia, lo recomendable es ubicar valores pequeños para mejorar la respuesta del controlador, ver Figura 47.

Nota: Optar por un valor de cero en el peso de salida causará que la misma no se mantenga en el Set-point seleccionado. Está opción es viable solamente si se desea que funcione como una variable indicadora.

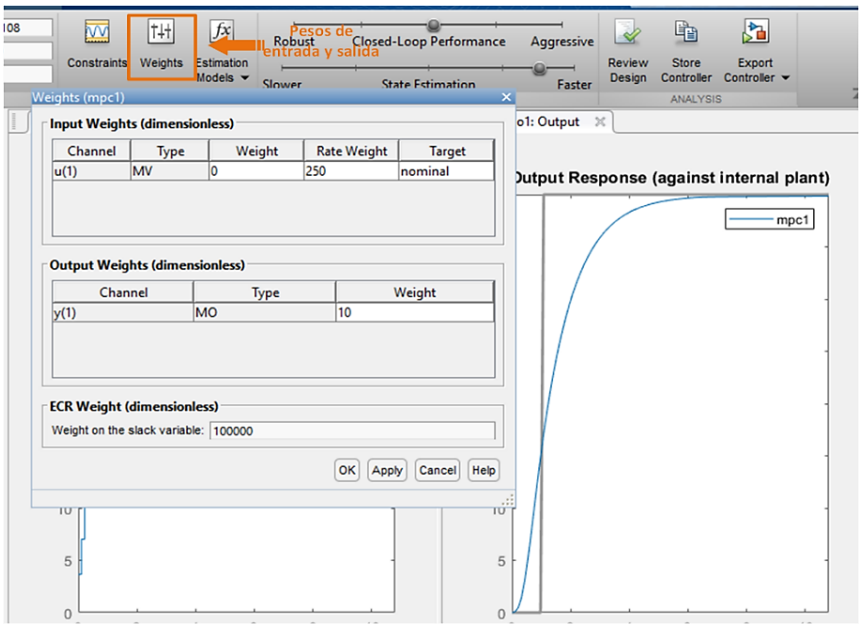


Figura 47. Pesos colocados a la entra y salida del modelo

Para finalizar la parte de sintonización del modelo se puede modificar el rendimiento del controlador en lazo cerrado y velocidad de estimación del modelo, mientras el indicador del slider esté ubicado más a la derecha, el comportamiento del controlador será más agresivo y viceversa.

Nota: Mientras más agresivo sea el comportamiento del controlador, existirá mayor probabilidad de sobreimpulso a la salida.

4. Al finalizar la sintonización se carga el modelo al bloque MPC de simulink, véase la Figura 48.

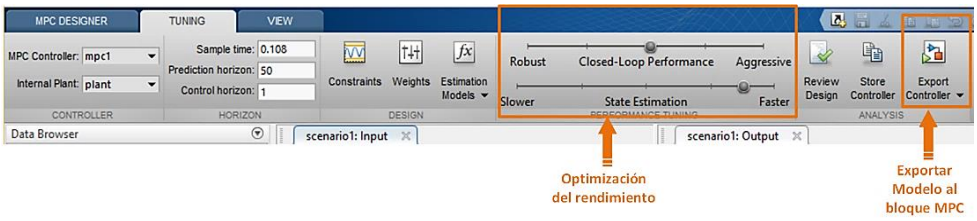


Figura 48. Exportar modelo al bloque MPC en Simulink

3.6.5. Simulación del modelo MPC

Con la sintonización realizada se procede a realizar simulaciones del proceso, la cual permite conocer cómo reacciona el sistema al ingresar cambios en el SP, también ayudará a observar cuál de los parámetros se adapta mejor al modelo para posteriormente implementarlo de forma real. Las simulaciones realizadas arrojan la siguiente información:

Si el horizonte de predicción es pequeño, provoca que el controlador actúe de forma lenta al cambio en la referencia, esto en términos prácticos podría compararse a un control clásico PID, véase la Figura 49.

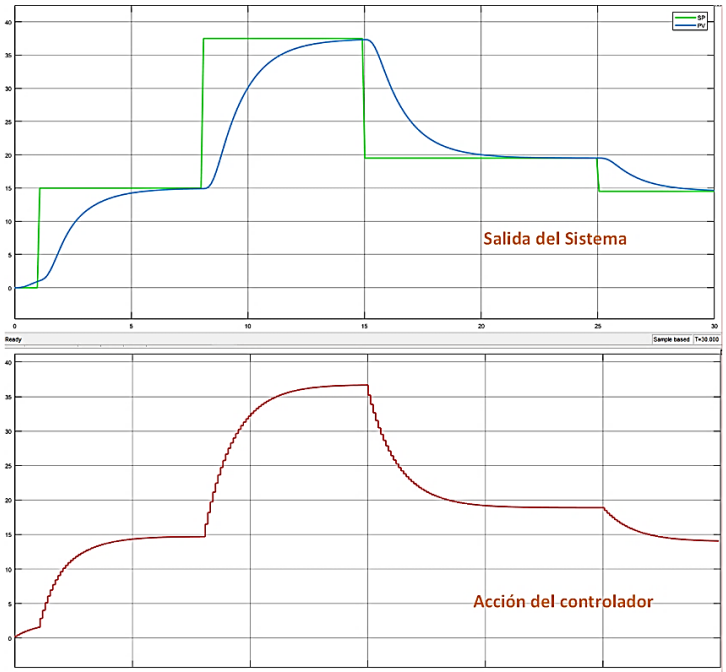


Figura 49. Respuesta del controlador con Hp pequeño

Cuando el horizonte de predicción es grande, el controlador actúa más rápido, es decir se incrementa la capacidad de predicción del controlador MPC, véase la Figura 50.

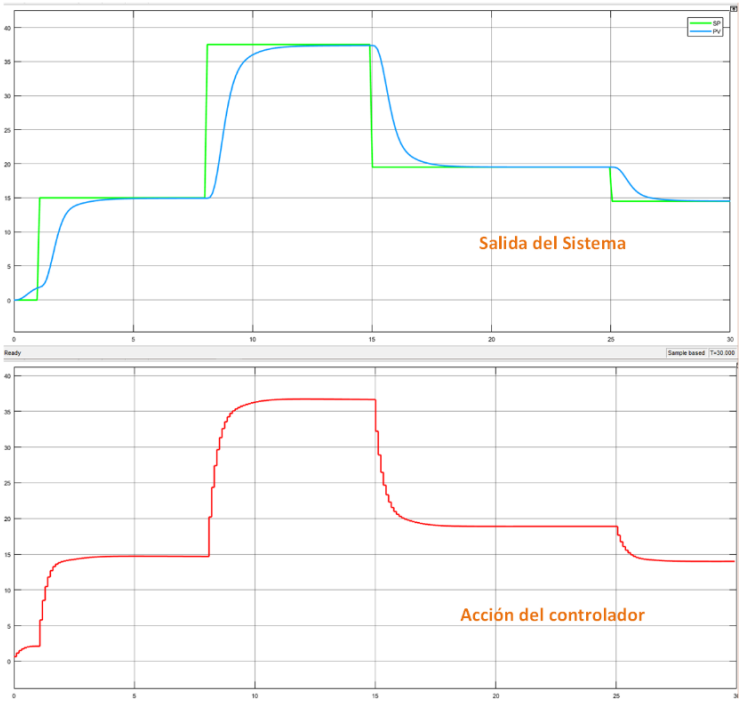


Figura 50. Respuesta del controlador con Hp grande

Sin ubicar pesos a la entrada y a la salida el controlador deja de seguir a la referencia, es decir pierde la capacidad de predecir un cambio a la salida, tal como se muestra en la Figura 51.

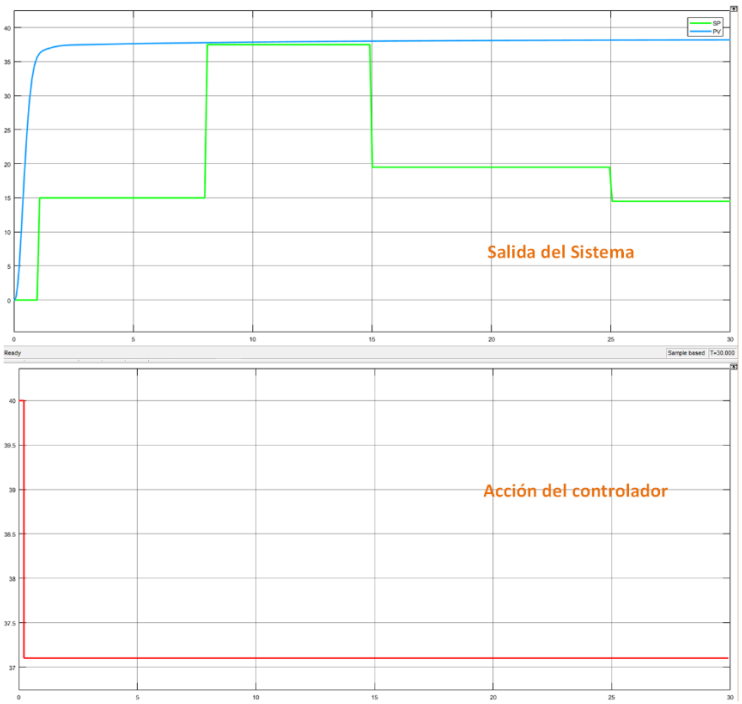


Figura 51. Respuesta del controlador sin pesos de entrada y salida

CAPÍTULO IV

4. IMPLEMENTACIÓN DE LOS CONTROLADORES

En este capítulo se describe la implementación de la propuesta. A nivel de hardware se presenta el tratamiento de las señales de entrada y salida que se ve plasmado en la elaboración de placas electrónicas. A nivel de software los acondicionamientos realizados, la programación del controlador borroso y del controlador predictivo por modelo. Además, el desarrollo de las interfaces de visualización, donde se puede evidenciar que los controladores operan de forma autónoma al monitorear en tiempo real el estado de las variables presentes en el proceso.

4.1. Implementación en hardware

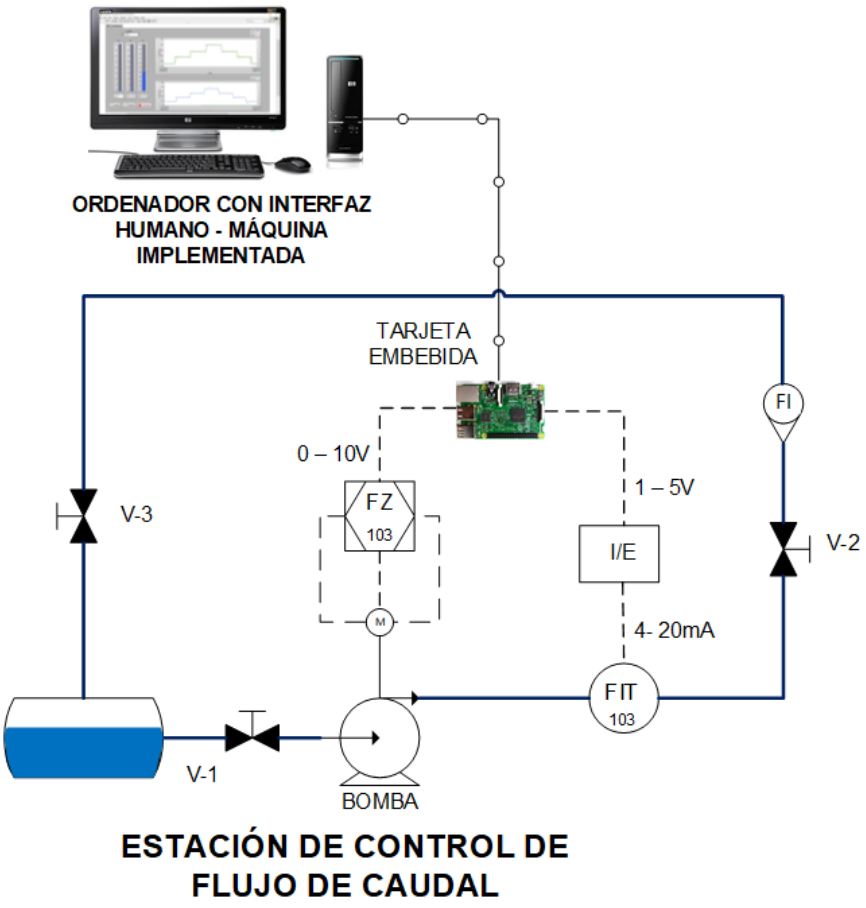


Figura 52. **Diagrama general del sistema de control de flujo de caudal**

En la Figura 52 se muestra el diagrama general del sistema de control de flujo de caudal, como ya se había mencionado en el capítulo anterior, no se usará la pantalla touch de dicha estación, sino que se desarrollará un HMI en un computador portátil y en lugar de utilizar su PLC, se utilizarán tarjetas embebidas de bajo costo. Para que éstas ofrezcan prestaciones similares es necesario que se realice un tratamiento previo de la señal que ingresa a la tarjeta y se proceda igual manera con la señal que sale de la misma. A continuación, se muestra dicho proceso:

4.1.1. Acondicionamiento de la señal previa al controlador

La señal estándar con la que trabaja la mayoría de los transmisores industriales es de 4 a 20 mA, pero las tarjetas que serán utilizadas lo hacen con señales de voltaje. Para ello es necesario implementar un convertidor de corriente a voltaje (I/E), lo cual se obtiene colocando una resistencia conectada en serie con el proceso. De esta forma, ahora se opera con voltaje. Se han utilizado dos rangos: el primero será de *1* a *5* voltios (*V*) y el segundo de *0.66* a *3.3V*, los valores de resistencias respectivos se calculan a partir de la ecuación (4.1) y (4.2), como se muestra:

$$V = I * R \tag{4.1}$$

$$R = \frac{V}{I} \tag{4.2}$$

$$R_5 = \frac{5V}{20mA} \qquad R_{3.3} = \frac{3.3V}{20mA}$$

$$\boldsymbol{R_5 = 250\Omega} \qquad \boldsymbol{R_{3.3} = 165\Omega}$$

Estos son valores ideales, ya que en la práctica dichos valores varían un poco. Adicionalmente, se colocó un capacitor electrolítico de $0.1\mu F$ entre los terminales positivo y negativo de la fuente de alimentación de *5V*.

4.1.1.1. Filtrado de la señal

Para eliminar señales de ruido de alta frecuencia no deseadas se realiza un filtro pasa-bajo RC con $f_{corte} = 100\ Hz$ y se asume $C = 1\mu F$, para calcular el valor de *R* se utiliza la ecuación (4.3).

$$f_{corte} = \frac{1}{2\pi R1C} \tag{4.3}$$

$$R1 = \frac{1}{2\pi fC}$$

$$R1 = \frac{1}{2\pi(100)(1\mu)}$$

$$\boldsymbol{R1 = 1.6\ K\Omega}$$

Siendo $1.5\ K\Omega$ el valor comercial más cercano, se optó por éste. Tomando en cuenta que se requiere eliminar señales de muy alta frecuencia, dicho cambio no afecta al diseño del filtro, pero se calcula el nuevo valor de la frecuencia de corte obtenida con estos elementos:

$$f_{corte} = \frac{1}{2\pi(1.5K)(1\mu)}$$

$$\boldsymbol{f_{corte} = 106.1\ Hz}$$

4.1.1.2. Acople de impedancias

Al trabajar con instrumentos industriales dentro del proceso que se requiere controlar, se ha evidenciado que las tarjetas ejercen efecto de carga y la señal de control no produce el efecto deseado, por lo cual se ha desarrollado un circuito seguidor de tensión con el amplificador de alta impedancia C.I. LF353 posterior al filtro.

4.1.1.3. Adquisición de datos

Según las características técnicas de las tarjetas, únicamente Udoo Neo Full posee un convertidor analógico – digital (ADC) interno con resolución de 16bits a 3.3V. Por su parte Beaglebone Black soporta 1.8V y Arduino 5V con resolución de 10 bits, mientras que Raspberry Pi en ninguna de sus versiones ofrece un ADC interno. Dado esto, con el objetivo de homologar este parámetro, en el caso de Udoo Neo Full se utilizará su propio convertidor por su alta resolución (utilizando solo 12 bits), pero en el caso de las demás tarjetas se ha seleccionado el circuito integrado (C.I.) MCP3202, un

convertidor de 12 bits de resolución de comunicación SPI, cuyo código digital de salida (DOC en inglés) se calcula con la ecuación (4.4)

$$DOC = \frac{4096 * V_{IN}}{V_{DD}} \tag{4.4}$$

Donde $V_{DD} = 5V$ y el valor de V_{IN} cambiará dependiendo del valor de la variable de proceso. En la Figura 53 se muestran las sub etapas que forman parte del acondicionamiento de la señal previo al controlador.

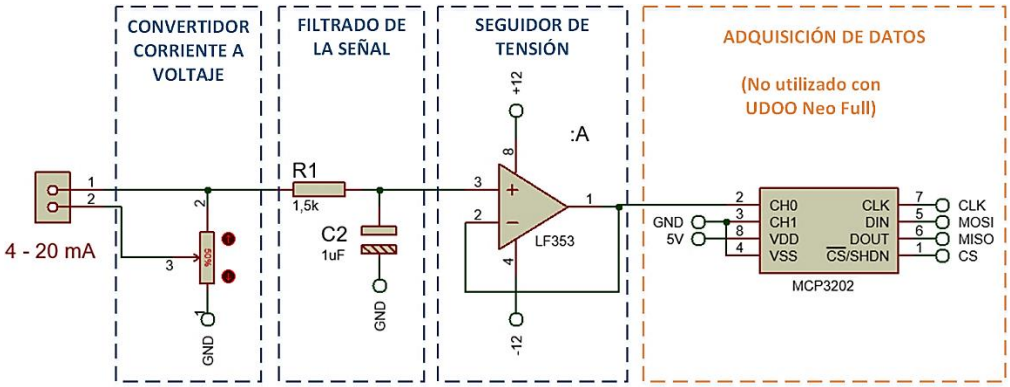


Figura 53. Diagrama del circuito electrónico previo a la etapa de control

4.1.2. Acondicionamiento de la señal posterior al controlador

4.1.2.1. Convertidor digital – analógico (DAC)

Una vez que se revisó las características técnicas de las tarjetas a utilizarse, se determinó que ninguna de ellas posee un DAC propio. Razón por la que se utilizó el convertidor externo C.I. PCF8591 con una resolución de 8 bits y comunicación I^2C, cuya salida se determina a partir de la ecuación (4.5).

$$V_{OUT} = V_{GND} + \frac{V_{REF} - V_{GND}}{256} \tag{4.5}$$

Donde $V_{GND} = 0V$ y $V_{REF} = 5V$. Entonces la variación de voltaje por cada valor entre 0 y 255 es de:

$$V_{OUT} = 0V + \frac{5V - 0V}{256}$$

$$\boldsymbol{V_{OUT} = 0.01953125V}$$

4.1.2.2. Acople de impedancias y amplificación de la señal

La señal que llegará al actuador (variador de frecuencia) que opera en un rango de voltaje de *0* a *10 V*, también debe pasar por un circuito seguidor de tensión para evitar efectos de carga en el proceso.

El voltaje máximo de la señal de control viene dado por el V_{REF} del DAC externo utilizado que es igual a *5V*, para lo cual fue necesario implementar un circuito amplificador no inversor de ganancia 2 como se muestra en la ecuación (4.6) y así cumplir con los requerimientos del variador. El C.I. LF353 posee dos amplificadores operacionales en el mismo circuito integrado permitiendo así optimizar recursos.

$$V_{OUT} = \left(\frac{R6}{R5} + 1\right) * V_{IN} \tag{4.6}$$

Donde, $R6 = 10K\Omega$ y $R5 = 10K\Omega$, obteniéndose:

$$V_{OUT} = \left(\frac{10K\Omega}{10K\Omega} + 1\right) * V_{IN}$$

$$V_{OUT} = (1 + 1) * V_{IN}$$

$$\boldsymbol{V_{OUT} = 2 * V_{IN}}$$

En la Figura 54 se muestra todas las sub etapas que forman parte del acondicionamiento de señales posterior a la etapa de control.

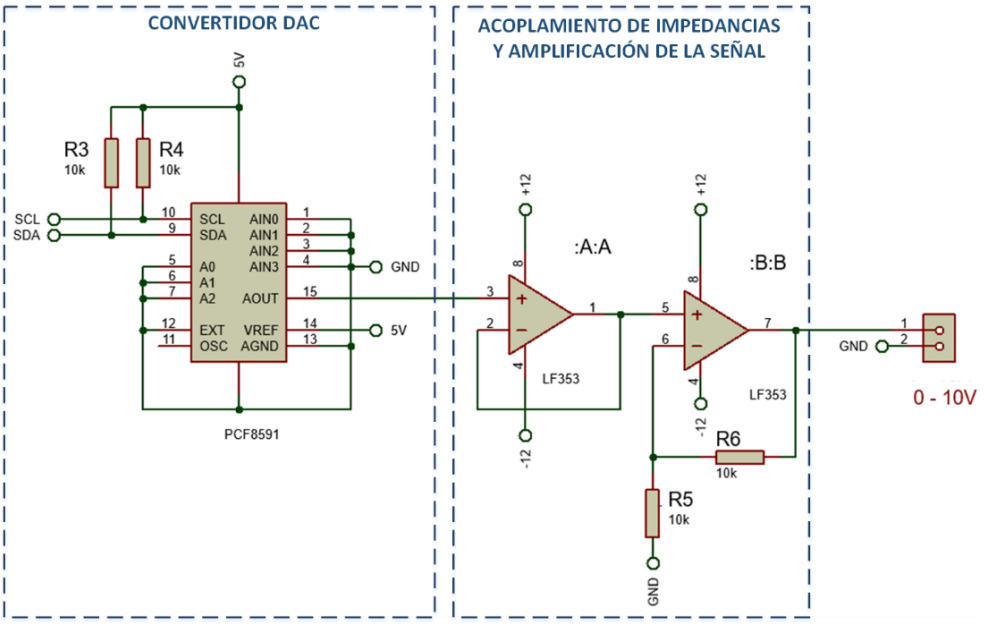


Figura 54. Diagrama del circuito electrónico posterior a la etapa de control

4.1.3. Elaboración de placas electrónicas

Utilizando el software PROTEUS versión 8.0, mediante su herramienta ISIS se ha realizado el diseño del circuito electrónico que se muestra en la Figura 55, en el cual se muestran todos los componentes electrónicos que han sido utilizados. Además, a través de la herramienta PCB Layout se realizó el diseño de las placas electrónicas, en la Figura 56 se presenta el diagrama de la placa realizada para la tarjeta Arduino UNO y las restantes se incluyen en el anexo A.

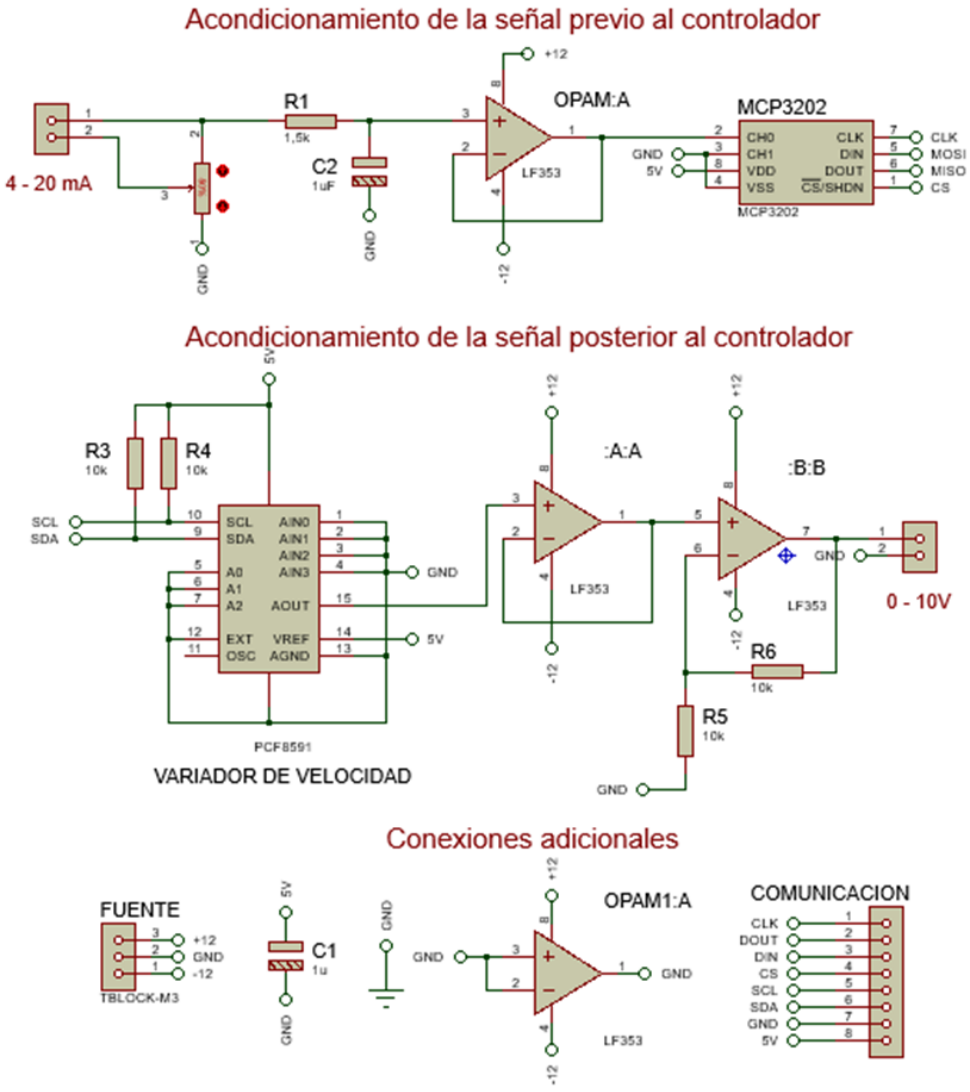


Figura 55. Diseño del diagrama del circuito electrónico desarrollado en la herramienta ISIS de Proteus

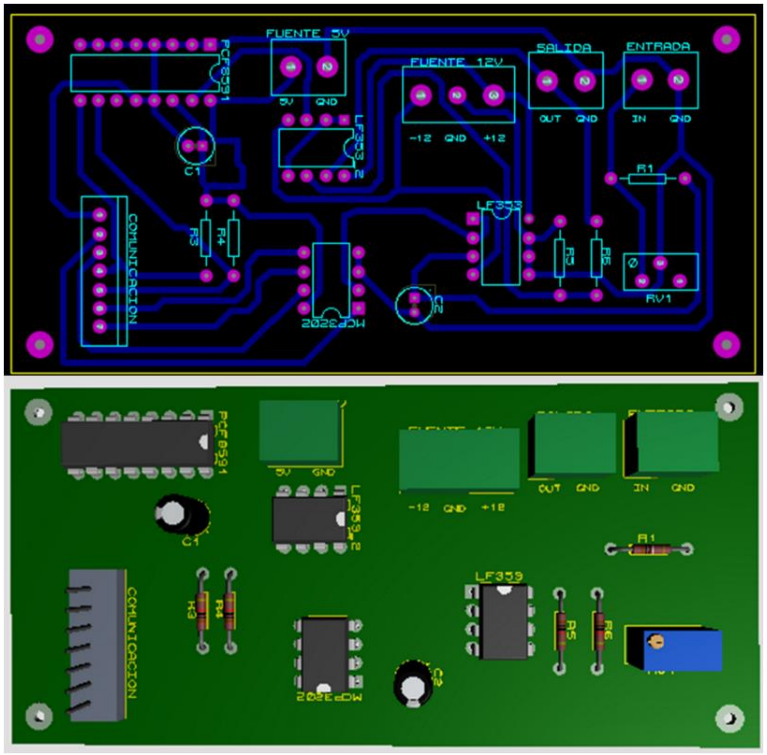


Figura 56. Diseño de la placa electrónica desarrollada en la herramienta PCB Layout de Proteus

4.2. Implementación de software

4.2.1. Escalamientos de variables

Posterior a la adquisición de datos de la variable de proceso se debe realizar un escalamiento de los valores obtenidos, para así poder trabajar numéricamente en los algoritmos de control correspondientes.

Para la adquisición de datos de las tarjetas Arduino UNO, Raspberry Pi 3y Beaglebone Black se utiliza el conversor MCP3202 que trabaja a 12 bits, es decir su rango es de 0 a 4095 cuando el voltaje que recibe varía de 0 a 5V respectivamente. Lo mismo que ocurre con el ADC interno de Udoo Neo Full. En la práctica éstos no son ideales, con una fuente de 5.02V los valores de voltaje provenientes de la estación de flujo de caudal son los siguientes:

$$V_{10\ LPM} = 0.972V$$

$$V_{40\ LPM} = 4.948V$$

Por lo tanto:

$$N_{10\ LPM} = \frac{0.972 * 4095}{5.02} = 792.9 \approx \mathbf{793}$$

$$N_{40\ LPM} = \frac{4.948 * 4095}{5.02} = 4036.27 \approx \mathbf{4036}$$

La ecuación de salida se obtiene a partir de (4.8), para lo cual previamente se determina el valor de la pendiente de la recta, con la ecuación (4.7)

$$m = \frac{y_2 - y_1}{x_2 - x_1} \tag{4.7}$$

$$m = \frac{150 - 0}{4036 - 793} = 0.04625346$$

$$y - y_1 = m(x - x_1) \tag{4.8}$$

$$y - 0 = m(x - 793)$$

$$\boldsymbol{y = 0.04625346x - 34.45820433}$$

Cuando se implementa el controlador MPC en Raspberry Pi3, se utiliza Arduino UNO como una tarjeta de adquisición de datos y dado que su ADC interno es de 10 bits, es decir opera con valores entre 0 y 1023 al variar el voltaje entre 0 y 5V, pero en la práctica se obtienen estos valores:

$$N_{10\ LPM} = \frac{0.972 * 1023}{5.02} = 198.08 \approx \mathbf{198}$$

$$N_{40\ LPM} = \frac{4.948 * 1023}{5.02} = 1008.33 \approx \mathbf{1008}$$

Para este escalamiento se tiene como entrada un rango de 198 a 1008 y como salida: 0 a 255, con lo cual se realizan los siguientes cálculos:

$$m = \frac{255 - 0}{1008 - 198} = 0.314814815$$

$$y - y_1 = m(x - x_1)$$

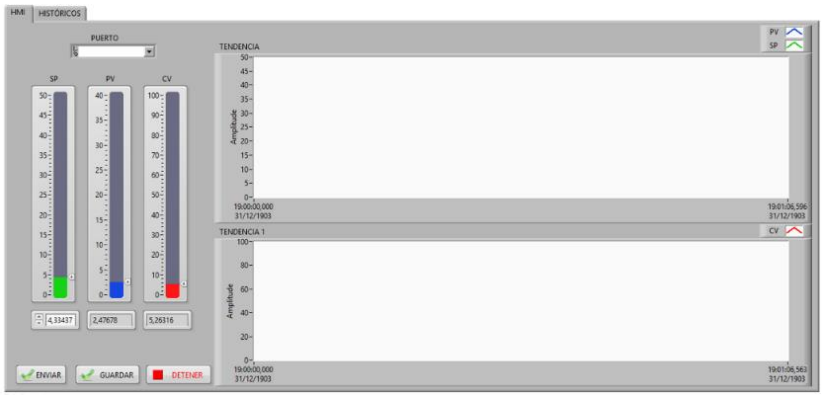
$$y - 0 = m(x - 198)$$

$$\boldsymbol{y = 0.314814815x - 62.33333333}$$

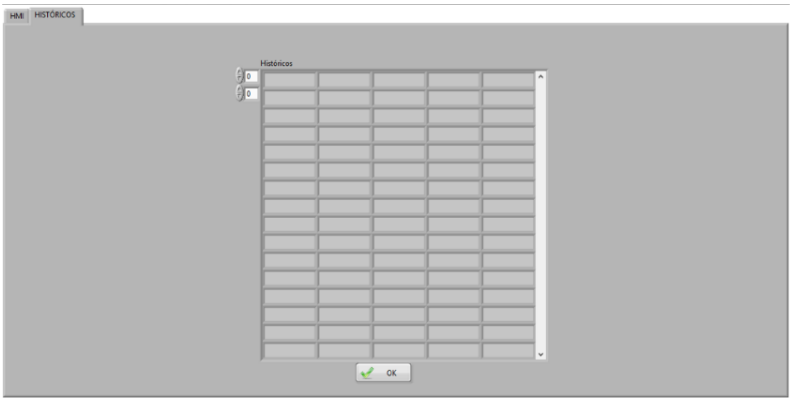
4.2.2. Implementación del HMI diseñado en LabVIEW

El panel frontal del HMI consta de tres indicadores tipo barra de progreso que corresponden al Set-point (SP), Variable de Proceso (PV) y la Variable de Control (CV); simbolizadas de color verde, azul y rojo respectivamente, como se muestra en la Figura 57; la interfaz también incluye un campo de entrada para elegir el puerto de comunicación (esto dependerá de la tarjeta que se use), además se incluye tres botones:

- **ENVIAR:** Este botón permite enviar el SP hacia el controlador implementado en las tarjetas embebidas, la razón de incluir este botón es debido a que la comunicación serial es de pregunta y respuesta, con esto se evita recibir datos basura en la trama que llega al HMI.
- **GUARDAR:** Este botón permite iniciar el proceso de guardado de datos, lo cuales se pueden verificar en la pestaña de Históricos.
- **CERRAR:** Se detiene la ejecución del HMI.



a)



b)

Figura 57. Panel frontal del HMI: a) Pestaña de tendencias b) Pestaña de históricos

En la pestaña de “Históricos” se encuentra la pantalla de presentación de los datos que se están guardando, además se coloca un botón que permite exportar los datos hacia una hoja de cálculo de Excel.

El diagrama de bloques del HMI está estructurado en 3 partes (véase la Figura 58), las cuales se explica a continuación:

- **Variables del controlador del proceso:** se hace escalamientos de las variables del proceso (SP, PV y CV) necesarios para la visualización en las tendencias del panel frontal.
- **Comunicación Serial:** En esta sección del diagrama de bloques lo que se hace primero es la escritura del SP y luego la lectura y separación de la trama que envía el puerto serial de las tarjetas embebidas; de esta manera se logra tomar los valores de cada variable del proceso y dirigirlas hacia los indicadores (barras de progreso y tendencias) del panel frontal.
- **Guardar y Exportar datos a Excel:** Finalmente en este bloque lo que se hace es guardar los datos de las tendencias, este proceso se ejecuta una vez que se presiona el botón “GUARDAR”, el bloque también permite exportar los datos hacia Excel.

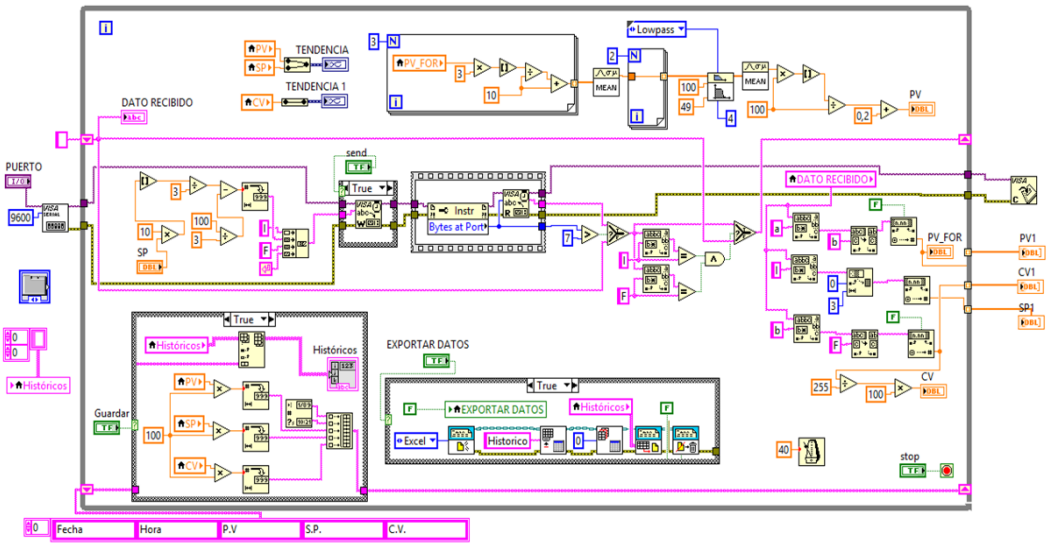


Figura 58. Diagrama de bloques del HMI

4.2.3. Implementación del Control Borroso en las tarjetas embebidas de bajo costo

El código usado para el emborronado, la elaboración de la base de reglas y el desemborronado son la base del control en todas las tarjetas usadas (Arduino UNO, Raspberry Pi3, Beaglebone Black, UDOO Neo Full). La diferencia en la implementación del programa entre cada una de las tarjetas se menciona a continuación:

- La librería usada para implementar el control borroso, está orientada para el IDE de Arduino (a pesar que su creador especifica que la librería es válida para cualquier tarjeta embebida que use código C++/C), por esta razón Arduino como Udoo Neo full presentan variaciones solo en la declaración de su comunicación serial y la cual se detalla de mejor manera en el Anexo B.1 para el primer caso y Anexo B.3 para el segundo.
- Para el caso de Raspberry Pi 3 y Beaglebone Black el programa se debió migrar a un código C++, el cual se encuentra detallado en el Anexo B.2.
- En el caso de Raspberry Pi 3, Beaglebone Black y Udoo Neo full es necesario habilitar los puestos seriales para establecer la comunicación con el HMI desarrollado en LabVIEW, esto en Arduino es más sencillo debido a que la comunicación se realiza a través del cable de alimentación.

4.2.3.1. Inclusión de librerías

Se incluye todas librerías del controlador borroso, junto con librerías destinadas al funcionamiento de los conversores ADC y DAC externos, usadas para la adquisición o envío de datos según la tarjeta lo requiera, esto se observa en la Figura 59.

```
1  // DECLARACIÓN DE LIBRERÍAS
2    #include "Wire.h"
3    #include <SPI.h>
4    #include <MCP3202.h>
5    MCP3202 adc(10);
6    #include <FuzzyRule.h>
7    #include <FuzzyComposition.h>
8    #include <Fuzzy.h>
9    #include <FuzzyRuleConsequent.h>
10   #include <FuzzyOutput.h>
11   #include <FuzzyInput.h>
12   #include <FuzzyIO.h>
13   #include <FuzzySet.h>
14   #include <FuzzyRuleAntecedent.h>
```

a)

```
 8 #include <wiringPi.h>
 9 #include <wiringPiI2C.h>
10 #include <pcf8591.h>
11 #include <wiringSerial.h>
12 #include "../../FuzzyRule.h"
13 #include "../../FuzzyComposition.h"
14 #include "../../Fuzzy.h"
15 #include "../../FuzzyRuleConsequent.h"
16 #include "../../FuzzyOutput.h"
17 #include "../../FuzzyInput.h"
18 #include "../../FuzzyIO.h"
19 #include "../../FuzzySet.h"
20 #include "../../FuzzyRuleAntecedent.h"
21 #include "../../mcp3008Spi.h"
22
```

b)

Figura 59. Inclusión de librerías en: a) Arduino, b) Raspberry Pi 3

4.2.3.2. Declaración de variables

Se declara variables que sirven para el desarrollo del controlador, variables que ayudan a la comunicación serial entre la tarjeta y el HMI en LabVIEW y una variable propia que sirve para instanciar un objeto tipo fuzzy, véase la Figura 60.

```
16 // VARIABLES DE CONTROL
17   float Pv = 0.0;
18   float Sp = 0.0;
19   float Cv = 0.0;
20   float Error = 0.0;
21   float Salida = 0.0;
22   float Aux3 = 0.0;
23   byte Actuador = 0;
24
25 // INSTANCIAR UN OBJETO TIPO FUZZY
26   Fuzzy* fuzzy = new Fuzzy();
27
```

a)

```
using namespace std;

float Pv=0.0;
double Sp = 20.0;
float Pv1 = 0.0;
float Sp1 = 0.0;
float Cv = 0.0;
double Escala=0.0;
float error = 0.0;
float dif = 0.0;
float output = 0.0;
float x = 0.0;
float y = 0.0;
Fuzzy* fuzzy = new Fuzzy();
```

b)

Figura 60. Declaración de variables en: a) Arduino, b) Raspberry Pi3

4.2.3.3. Creación de los conjuntos borrosos para el Emborronado

Como se mencionó anteriormente el código del control borroso es la base para todas las tarjetas (véase Figura 61), en el programa se usa las siguientes declaraciones:

- **FuzzySet:** Sirve para modelar el sistema
- **FuzzyInput:** Agrupa todas las entradas que formen el emborronado.
- **addFuzzySet:** Añade el conjunto borroso en este caso de entrada.

```
// ALGORITMO DE CONTROL BORROSO
// Agrupa toddos los conjuntos borrosos  de entrada que pertenecen al mismo dominio.
  FuzzyInput* err_caudal = new FuzzyInput(1);

// Conjuntos borrosos de entrada
  FuzzySet* NA = new FuzzySet(-100,-100, -75, -50); // Error Negativo Alto
  err_caudal->addFuzzySet(NA);
  FuzzySet* NM = new FuzzySet(-75,-50, -50, -25); // Error Negativo Medio
  err_caudal->addFuzzySet(NM);
  FuzzySet* NB = new FuzzySet(-50,-25, -25, 0); // Error Negativo Bajo
  err_caudal->addFuzzySet(NB);
  FuzzySet* CE = new FuzzySet(-25,-0,0,25); // Error Neutro
  err_caudal->addFuzzySet(CE);
  FuzzySet* PB = new FuzzySet(0,25,25,50); // Error Positivo Bajo
  err_caudal->addFuzzySet(PB);
  FuzzySet* PM = new FuzzySet(25,50,50,75); // Error Positivo Medio
  err_caudal->addFuzzySet(PM);
  FuzzySet* PA = new FuzzySet(50,75,100,100); // // Error Positivo Alto
  err_caudal->addFuzzySet(PA);
  fuzzy->addFuzzyInput(err_caudal); //Adiciona los conjuntos de entrada al objeto Borroso
```

Figura 61. Conjuntos borrosos de entrada

4.2.3.4. Creación de los conjuntos borrosos para el Desemborronado

Para los conjuntos borrosos del desemborronado (véase Figura 62) se implementa el método:

- **FuzzyOutput:** Este método agrupa todos los conjuntos borrosos de salida.

Los demás métodos que se usan son exactamente similares a los que se usa para el emborronado.

```
// Agrupa toddos los conjuntos borrosos  de entrada que pertenecen al mismo dominio.
  FuzzyOutput* Voltaje = new FuzzyOutput(1);
// Adicionando conjuntos borrosos de salida
  FuzzySet* Vmuyb = new FuzzySet(-33.33,-33.33,-10,-5.7); // Voltaje Muy Bajo
  Voltaje->addFuzzySet(Vmuyb);
  FuzzySet* Vb = new FuzzySet(-9.033, -5.7, -5.7,-2.833); // Voltaje Bajo
  Voltaje->addFuzzySet(Vb);
  FuzzySet* Vmb = new FuzzySet(-5.7,-2.833,-2.833,0); // Voltaje Medio Bajo
  Voltaje->addFuzzySet(Vmb);
  FuzzySet* C = new FuzzySet(-2.833, 0, 0, 2.833); // Voltaje Cero
  Voltaje->addFuzzySet(C);
  FuzzySet* Vma = new FuzzySet(0,2.833,2.833,5.7); // Voltaje Medio Alto
  Voltaje->addFuzzySet(Vma);
  FuzzySet* Va = new FuzzySet(2.833,5.7,5.7,9.033); // Voltaje Alto
  Voltaje->addFuzzySet(Va);
  FuzzySet* Vmuya = new FuzzySet(5.7,10,10,10); // Voltaje Muy Alto
  Voltaje->addFuzzySet(Vmuya); // Adicionando o FuzzySet T
  fuzzy->addFuzzyOutput(Voltaje); // Adicionando o FuzzyOutput no objeto Fuzzy
```

Figura 62. Conjuntos difusos de salida

4.2.3.5. Desarrollo de la base de reglas borrosas

Para implementar toda la base de conocimiento del controlador borroso, la librería hace uso de la inferencia tipo Mamdani, tanto para realizar la base de datos como para la base de reglas del comportamiento del sistema (véase Figura 63); los métodos empleados son los siguientes:

- **FuzzyRuleAntecedent:** Este método enlaza el antecedente de la expresión condicional de la regla borrosa
- **joinSingle:** Se usa para construir una expresión simple (En este caso solo se tiene una entrada y una salida).
- **FuzzyRuleConsequent:** Enlaza la salida de la expresión condicional de la regla borrosa.
- **FuzzyRule:** Esta función es usada para montar la regla borrosa del controlador con el antecedente y consecuente mencionado anteriormente.
- **addFuzzyRule:** Este método añade la regla borrosa del controlador.

```
88 // Base de Reglas: Inferencia de Mamdani
89   FuzzyRuleAntecedent* SI_NA = new FuzzyRuleAntecedent(); //Instanciando un Antecedente
90   SI_NA->joinSingle(NA); // Adicionando un Conjunto de Entrada al Antecedente
91   FuzzyRuleConsequent* ENTONCES_Vmuyb = new FuzzyRuleConsequent(); // Instanciando un Cc
92   ENTONCES_Vmuyb->addOutput(Vmuyb); // Adicionando un Conjunto de Salida al Consecuente
93
94   FuzzyRuleAntecedent* SI_NM = new FuzzyRuleAntecedent();
95   SI_NM->joinSingle(NM);
96   FuzzyRuleConsequent* ENTONCES_Vb= new FuzzyRuleConsequent();
97   ENTONCES_Vb->addOutput(Vb);
98
99   FuzzyRuleAntecedent* SI_NB = new FuzzyRuleAntecedent();
100  SI_NB->joinSingle(NB);
101  FuzzyRuleConsequent* ENTONCES_Vmb = new FuzzyRuleConsequent();
102  ENTONCES_Vmb->addOutput(Vmb);
103
104  FuzzyRuleAntecedent* SI_CE = new FuzzyRuleAntecedent();
105  SI_CE->joinSingle(CE);
106  FuzzyRuleConsequent* ENTONCES_C = new FuzzyRuleConsequent();
107  ENTONCES_C->addOutput(C);
108
109  FuzzyRuleAntecedent* SI_PB = new FuzzyRuleAntecedent();
110  SI_PB->joinSingle(PB);
111  FuzzyRuleConsequent* ENTONCES_Vma = new FuzzyRuleConsequent();
112  ENTONCES_Vma->addOutput(Vma);
123
124 // Instanciando las reglas borrosas
125   FuzzyRule* fuzzyRule01 = new FuzzyRule(1, SI_NA, ENTONCES_Vmuyb);//Instancia la regla
126   fuzzy->addFuzzyRule(fuzzyRule01); //Adiciona la regla al objeto Borroso
127   FuzzyRule* fuzzyRule02 = new FuzzyRule(2, SI_NM, ENTONCES_Vb);
128   fuzzy->addFuzzyRule(fuzzyRule02);
129   FuzzyRule* fuzzyRule03 = new FuzzyRule(3, SI_NB, ENTONCES_Vmb);
130   fuzzy->addFuzzyRule(fuzzyRule03);
131   FuzzyRule* fuzzyRule04 = new FuzzyRule(4, SI_CE, ENTONCES_C);
132   fuzzy->addFuzzyRule(fuzzyRule04);
133   FuzzyRule* fuzzyRule05 = new FuzzyRule(5, SI_PB, ENTONCES_Vma);
134   fuzzy->addFuzzyRule(fuzzyRule05);
135   FuzzyRule* fuzzyRule06 = new FuzzyRule(6, SI_PM, ENTONCES_Va);
136   fuzzy->addFuzzyRule(fuzzyRule06);
137   FuzzyRule* fuzzyRule07 = new FuzzyRule(7, SI_PA, ENTONCES_Vmuya);
138   fuzzy->addFuzzyRule(fuzzyRule07);
139 }
```

Figura 63. Reglas de la base de conocimiento del controlador borroso

4.2.3.6. Escalamiento para la adquisición de datos, emborronado y desemborronado del proceso

Debido a la resolución de 12 bits del ADC que usado para la adquisición de datos y el rango de trabajo amplio que se requiere para los conjuntos de emborronado y desemborronado, es necesario aplicar un escalamiento de la variable.

- En Arduino existe una librería propia para adquirir los datos desde el integrado MPC3202, la cual puede verse en la Figura 64.

```
141 void loop(){
142 //Escalamiento para la adquisición de datos
143   //Pv = (100/812.262)*analogRead(A1) - 25.1889168766;
144   float Pv0 = adc.read(0, 12);
145   Pv = (50.0/3230.0)*Pv0 - 12.260061919;
146   Error = Sp-Pv;
```

Lectura a 12 bits

Figura 64. Lectura a 12 bits y escalamiento de la variable de entrada en Arduino

- En Raspberry y Beaglebone Black la adquisición de datos del ADC se hace con una librería propia de los dispositivos, tal como se puede ver en la Figura 65.

```
229 mcp3008Spi a2d("/dev/spidev0.0", SPI_MODE_0, 1000000, 8);
230
231   data[0] = 1;  //  first byte transmitted -> start bit
232   data[1] = 0b10000000 |( ((a2dChannel & 7) << 4)); // secon
233   data[2] = 0; // third byte transmitted....don't care
234
235   a2d.spiWriteRead(data, sizeof(data) );
236
237   a2dVal = (data[1]<< 8) & 0b111100000000; //merge data[1] &
238   a2dVal |=  (data[2] & 0xff);
239
240   cout << "Lectura del ADC: " << a2dVal << endl;
241
242  Escala = a2dVal;
246  Pv = (100/3195.852)*Escala - 24.5838668374; // 1 ... 5V
247 // Pv = (100/780)*Escala - 29.48717;
248  cout << "VALOR PV " << Pv << endl;
249  cout << "VALOR Sp " << Sp << endl;
250
251  error = Sp-Pv;
252  cout << "ERROR: " << error << endl;
253
```

Código para leer los datos a 12 bits

Figura 65. Lectura a 12 bits y escalamiento de la variable de entrada en Raspberry Pi3 y Beaglebone Black

- En Udoo Neo full la lectura a 12 bits se realiza con el ADC propio de la tarjeta, véase la Figura 66:

```
138 void loop(){
139 //Escalamiento para la adquisición de datos
140   analogReadResolution(12);
141   Pv = (100/3295.0)*analogRead(A1) - 24.2792109256;
142   Error = Sp-Pv;
143   Pv1 = Pv;
144   Sp1 = Sp;
145   Cv1 = Cv;
```

Lectura a 12 bits con el ADC propio de la tarjeta

Figura 66. Lectura a 12 bits y escalamiento de la variable de entrada en Udoo Neo Full

En la Figura 67 se puede observar el código que se implementa para el emborronado y desemborronado, los métodos que se describen a continuación son propios de la librería eFLL.

- **fuzzy -> setInput:** Pasa el valor a evaluar de la entrada al sistema.
- **fuzzify():** Inicia el proceso de emborronado.
- **defuzzify():** Termina el proceso del controlador borroso y arroja el valor de salida para enviar al controlador.

```
148 //EMBORRONADO Y DESEMBORRONADO
149   fuzzy -> setInput(1, Error);
150   fuzzy -> fuzzify();
151   Salida = fuzzy -> defuzzify(1);
152   Salida = Aux3+Salida;
153
154   if(Salida >= 50){
155     Salida = 50;
156   }
157   if(Salida <= -50){
158     Salida = -50;
159   }
160
161   Aux3 = Salida;
162   Cv = Salida*2.55+127.5;
163   Actuador = Cv;
```

Figura 67. Emborronado y desemborronado del proceso

4.2.3.7. Comunicación serial entre las tarjetas embebidas y el HMI en LabVIEW

Para evitar que las tarjetas envíen constantemente los datos por el puerto serial, se desarrolla un método de pregunta y respuesta entre el código cargado en las tarjetas y LabVIEW, como se ve en la Figura 68, se revisa que el buffer posea datos y mediante caracteres tipo string que indican el inicio y final de la trama se envía los datos cada vez que el puerto serial esté disponible, con esto se evita desincronización en los tiempo de envío y recepción de la comunicación serie.

```
// LECTURA DEL PUERTO SERIAL
  while(Serial.available() >= 1){
    Lectura = Serial.read();
    if(Lectura == 'F'){
      Recepcion += Lectura;
      aux = 0;
      count = 1;
      Recepcion2 = Recepcion;
      break;
    }
    if(Lectura == 'I'){
      Recepcion += Lectura;
      aux=1;
    }
    if(aux == 1){
      Recepcion += Lectura;
    }
  }

  if(count == 1){
    Bandera = 0;
    Recepcion2 = Recepcion;
    Longitud = Recepcion.length();
    n1 = Recepcion.substring(2,Longitud-1);
    Sp = n1.toFloat();// convierte el valor
    Recepcion = "";
    count = 0;
```

```
// ESCRITURA EN EL PUERTO SERIAL
  String D1 =  String(Sp,2);
  String D2 =  String(Pv,3);
  String D3 =  String(Cv,3);
  Cadena =  String('I'+D1+'a'+D2+'b'+D3+'F');
  Serial.println(Cadena);

// SALIDA AL ACTUADOR
  Wire.beginTransmission(0x90 >> 1);
  Wire.write(0x40);
  Wire.write(Actuador);
  Wire.endTransmission();

  delay(40);
}
```

a)

```
  cout  << "WELCOME TO RASP"<< endl;

 //Comunicacion Serial
 dataAv= serialDataAvail(fd);
 while (dataAv>=1){
   c = serialGetchar (fd) ;

 if(c=='F'){
  Recepcion += c;aux=0;count=1;Recepcion2=Recepcion;break;
 }

 if(c=='I'){
  Recepcion += c;aux=1;
 }

  if (aux==1){
   Recepcion += c;
  }

 }

  if (count == 1){
  flagRequest = 0;
  Recepcion2=Recepcion;
  x= ::Recepcion.length();
  n1= Recepcion.substr(2,x-1);
  cout <<"El Substring es: "<<n1<<endl;
  Sp = ::strtod(n1.c_str(),0);// convierte el valor de stri
  cout << Sp <<  endl;
  Recepcion = "";
  count=0;
 }
```

```
    cout << "Salida: " << output<< " \n  \n" << endl;
    stringstream conv1;
    stringstream conv2;
    stringstream conv3;
    conv1 << Pv;
    conv2 << Sp;
    conv3 << Cv;

    string str  = conv1.str();
    string str1 = conv2.str();
    string str2 = conv3.str();

    datos =I+str1+a+str+B+str2+F;
    cout << "DATOS \n  " << datos << endl;
    char* datos1 = (char*)datos.c_str();
   // const char c = datos.c_str();
    cout << "DATOS --> \n " << datos1 << " \n  \n " << endl;
    serialPuts(fd, datos1);
    serialFlush(fd);

    delay( 40 );
}
    return 0;
}
```

b)

Figura 68. Envío trama serial al HMI desarrollado en LabVIEW

La trama que se envía está compuesta de un carácter **“I”,** que indica el inicio de la trama, luego se envía el SP, seguido de un carácter **“a”,** que sirve como división entre los valores enviados, luego se envía el PV seguido de un carácter **“b”,** que de la misma forma sirve como división entre los valores de la trama, finalmente se envía un carácter **“F”,** para indicar la finalización de la trama.

4.2.4. Comunicación entre Simulink y la tarjeta Raspberry Pi 3

Antes de implementar el MPC en la tarjeta Raspberry Pi 3 es necesario descargarse los paquetes de soporte, tanto para MATLAB como para Simulink; la última versión del software y con la cual se desarrolló el control es MATLAB 2017ª, cabe recalcar que esta es la única versión donde ya viene incorporada los complementos para el modelo B de Raspberry Pi 3.

En la Figura 69 se puede observar los paquetes que se debe instalar para cargar el control MPC en la tarjeta, por otra parte la guía de instalación de los complementos se encuentran en el siguiente enlace proporcionado por MathWorks: https://es.mathworks.com/videos/install-the-matlab-support-package-for-raspberry-pi-94266.html, además que proporciona ejemplos para comprender mejor el funcionamiento de este paquete.

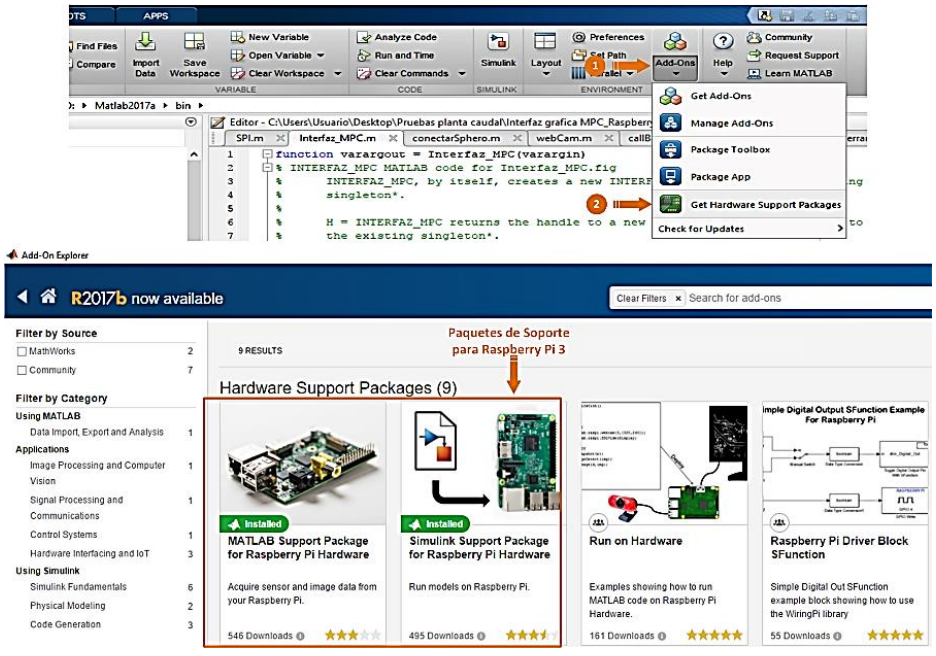


Figura 69. Paquetes de Raspberry Pi3 para instalar en MATLAB

Una vez que se complete la instalación se puede conectar la tarjeta embebida con Simulink, para eso es necesario seguir los siguientes pasos:

1. Conectar la tarjeta Raspberry Pi 3 mediante la dirección IP configurada al momento de la instalación de los paquetes de soporte, esto se logra mediante la siguiente función:

```
mypi=raspi('192.168.0.20','pi','raspberry')
```

   En la función anterior se específica dirección IP, usuario y contraseña.

2. En Simulink elegir el modo "External", ya que el modelo creado será cargado en un dispositivo externo al software, en este caso la tarjeta Raspberry Pi 3, una vez hecho esto entramos a la configuración de parámetros del modelo **(Model Configuration Parameters),** véase la Figura 70.

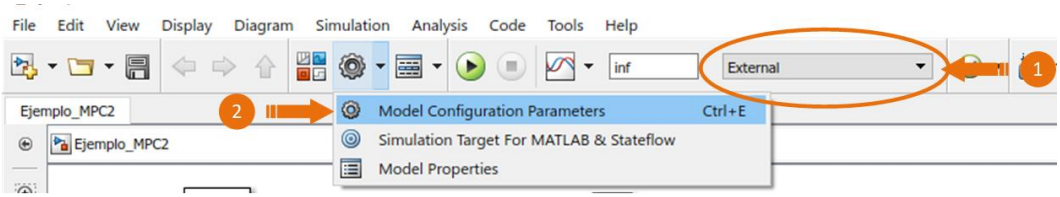


Figura 70. Conexión Raspberry Pi 3 con Simulink

3. En la ventana que se despliega se configuran los siguientes parámetros de la opción "Hardware Implementation" (véase Figura 71):

   - **45Hardware board:** Elige la tarjeta donde se vaya a cargar el modelo **(Raspberry Pi)**
   - **Board Parameters:** Muestra los parámetros de conexión de la tarjeta **(192.168.0.20; "pi"; "raspberry")**
   - **Build options:** Acciones que se realiza al construir el modelo sobre la tarjeta, entre ellas seleccionar el directorio de destino del modelo. **(home/pi/2ejemplo)**
   - **SPI:** Se configura la velocidad de la comunicación SPI en caso de usar estas opciones. **(No se usa para el control)**
   - **External mode:** Verifica la interface de comunicación y el puerto **(TCP/IP; 17725)**

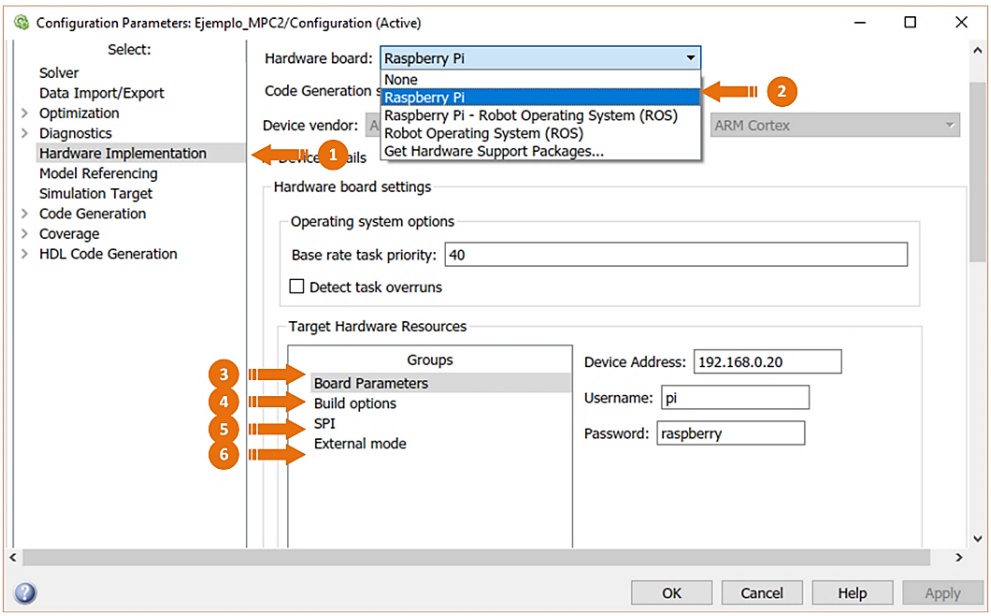


Figura 71. Configuración de parámetros para la conexión con Raspberry Pi 3

4.2.5. Implementación del control MPC en la tarjeta Raspberry Pi 3

Para implementar el controlador se diseñó una interfaz gráfica, usando el GUIDE de MATLAB. La pantalla principal consta de 5 botones, véase Figura 72:

- **Conectar Raspberry Pi3:** Este primer botón permite establecer la conexión con la tarjeta Raspberry Pi 3 mediante la función mencionada anteriormente.
- **Importar Datos MPC:** El segundo botón permite importar los datos al workspace, este paso es fundamental ya que el modelo desarrollado en simulink necesita de estos valores para correr el controlador, entre estas variables se tiene la planta que se importó en el diseño del modelo, restricciones del proceso, etc. En el Anexo B.4 se detallará el código usado para importar datos guardados previamente hacia la programación del guide.
- **Abrir Modelo MPC – Simulink:** Este botón permite llamar el diagrama de bloques desarrollado en simulink.

- **Guardar Datos Excel:** Este cuarto botón permite exportar los datos generados por el modelo de simulink hacia Excel y así obtener un mejor tratamiento de los mismos.
- **Cerrar:** Cierra la ventana principal y limpia el workspace.

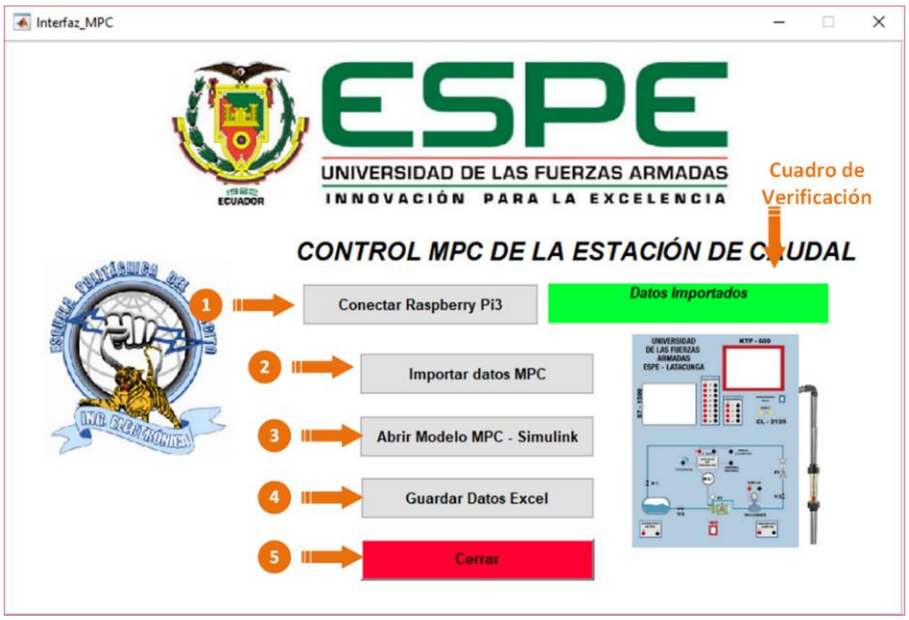


Figura 72. Ventana principal del Control MPC

En la Figura 73 se observa el diagrama de bloques diseñado para efectuar el control MPC; hay que recalcar que Raspberry Pi 3 carece de entradas analógicas, por este motivo se optó como solución usar una tarjeta Arduino UNO R3 como tarjeta de adquisición de datos y mediante un programa cargado en Arduino, se hace uso del puerto I2C para enviar los datos hacia el modelo en Simulink, el programa usado se detallará en el Anexo B.5.

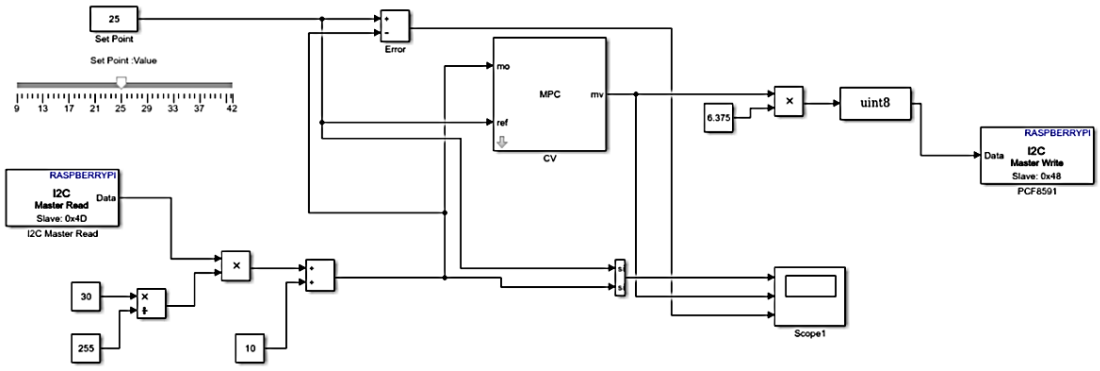


Figura 73. Diagrama de bloques del control MPC

4.2.4.1. Adquisición de datos

El paquete de soporte de Raspberry Pi 3 en la versión 2017, incorpora bloques para leer y escribir datos por comunicación I2C, en la Figura 74 se puede observar como después de adquirir los datos con el bloque “I2C Master Read”, se realiza un escalamiento de 10 – 40, el propósito de este escalamiento es enviar valores válidos para el MPC diseñado en el capítulo anterior.

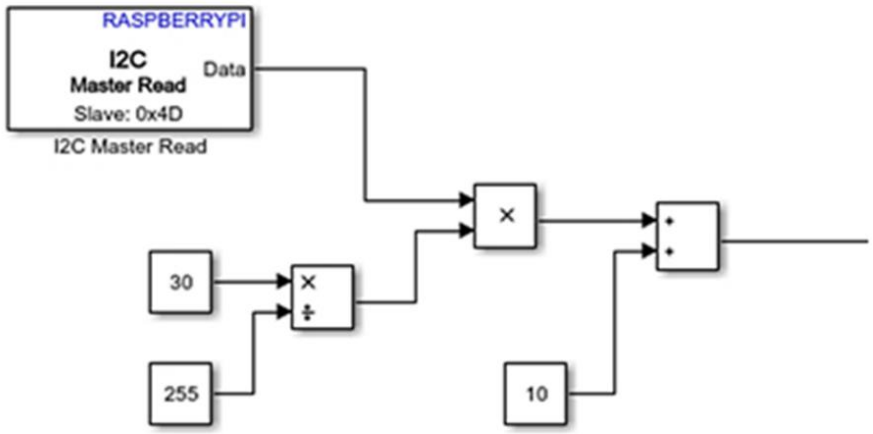


Figura 74. Adquisición de datos

El bloque “I2C Master Read”, como su nombre lo dice sirve para leer datos por medio de comunicación I2C; los parámetros que se debe configurar son los siguientes:

- **Board:** Elegir el modelo de la tarjeta **(Pi 3 Model B)**
- **Slave address:** Elegir la dirección del esclavo conectado **(4D = 77)**
  Nota: Esta dirección se obtiene fácilmente al ingresar por Putty a la tarjeta raspberry Pi 3 y se ejecuta el comando **“i2c-detect”.**
- **Data type:** Se elige el tipo de dato que se obtendrá a la salida del bloque **(uint8).**
- **Sample time:** Elegir el tiempo de muestreo **(0.108).**

4.2.4.2. Set-point y bloque de control

En la Figura 75 se puede observar el bloque del set-point y el bloque del control MPC, el cuál fue explicado en el capítulo de diseño y simulación; el set-point está asociado a un slider para hacer más interactivo el manejo de la referencia al momento de ejecutar el control.

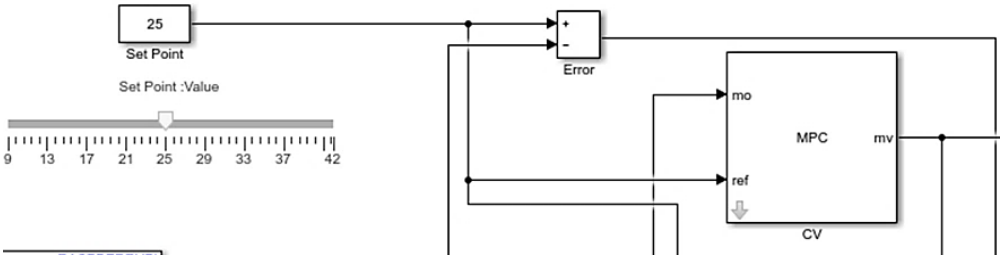


Figura 75. Bloque del Set-point y control MPC

4.2.4.3. Salida del controlador

A la salida del bloque de control se realiza un escalamiento previo de 0 – 255, debido a que el integrado que se usa para escribir **(PCF8591)** solo detecta valores de 8 bits $(2^8 = 255)$, luego de esta etapa se llega al bloque "I2C Master Write", como su nombre lo dice lo que hace es escribir valores de 8 bits para obtener a la salida del integrado voltajes de 0 – 5 V; los cuales son amplificados mediante un amplificador de alta impedancia y que en apartados anteriores se explicó el motivo de esta etapa, véase la Figura 76.

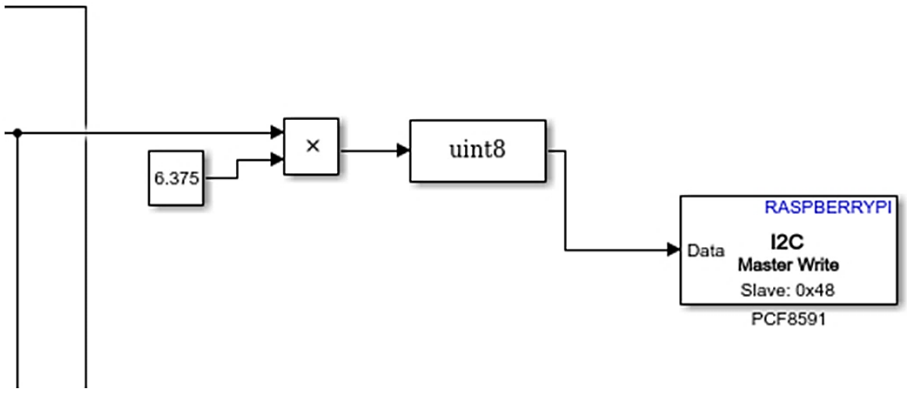


Figura 76. Salida del controlador

Los parámetros que se configuran en el bloque “I2C Master Write” son los siguientes:

- **Board:** Elegir el modelo de la tarjeta **(Pi 3 Model B)**
- **Slave address:** Seleccionar la dirección del esclavo por donde se va a escrbir los valores de 0 – 255 **(48 = 72).**

  Nota: La dirección de este esclavo como en el caso anterior se logra al ejecutar **“i2c-detect”** en la tarjeta.
- **Slave register address:** Se elige la dirección del registro del esclavo, esto se lo puede encontrar en las hojas de datos del integrado. **(120)**

CAPÍTULO V

5. RESULTADOS Y PRUEBAS EXPERIMENTALES

En este capítulo se describen los resultados que se han obtenido luego de la implementación de los controladores en tiempo real. Se mostrarán las curvas de respuesta de las variables del proceso, para realizar una comparación entre las tarjetas utilizadas y determinar en cual se evidenció un mejor desempeño.

El rango de operación de la planta de flujo de caudal es de 10 a 40 [LPM] y dentro de dicho intervalo se ha variado el valor de la consigna (SP) para evidenciar cuál es la respuesta de la variable de proceso (PV) ante dichos valores y además apreciar cómo varía la variable de control (CV) que determina la señal que se envía al actuador del proceso.

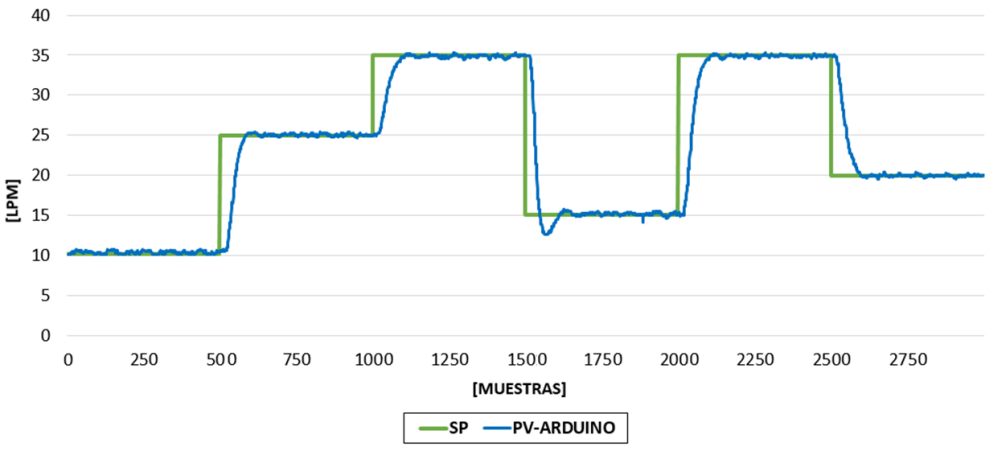
5.1. Análisis del controlador borroso en la planta de flujo de caudal

A continuación, se presenta el análisis individual de las curvas de respuesta que producen las variables del proceso, al implementar el algoritmo de control borroso en cada una de las tarjetas antes mencionadas. Para ello se inicia con el proceso en el límite inferior, es decir 10 LPM para luego modificar su valor a 25, 35, 15, 35 y 20 LMP cada 20 segundos respectivamente. Tomando en cuenta que los datos han sido guardados cada 40ms se tienen 25 muestras por segundo y por ende 500 muestras tras cada cambio de SP.

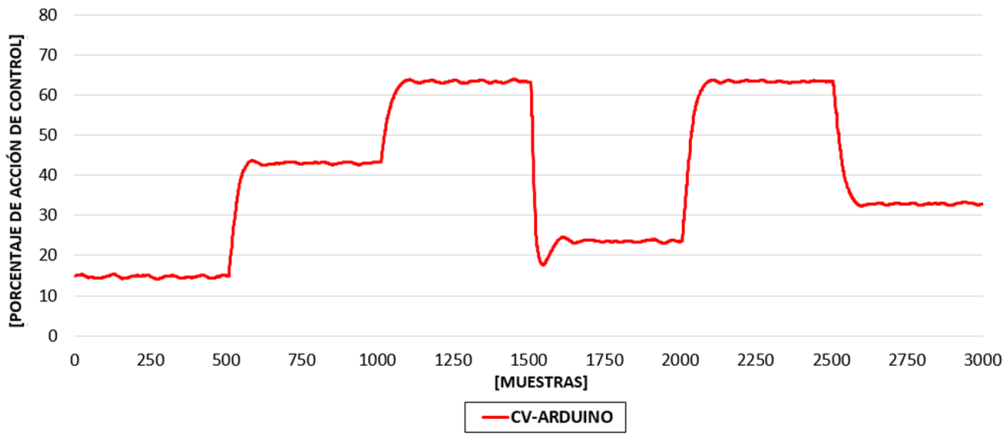
En las gráficas de SP vs. PV, en el eje vertical se encuentra el valor de litros por minuto y en el eje horizontal el número de muestras. Por otro lado, en la siguiente gráfica, se muestra en el eje vertical el porcentaje de acción de control y el eje horizontal de la misma forma el número de muestras tomadas en el tiempo.

5.1.1. Pruebas en la tarjeta Arduino UNO R3

En la Figura 77 a) se presenta la curva de respuesta SP vs. PV de Arduino UNO R3, donde al momento de realizar el cambio de SP de 35 a 15 LPM se nota sobreimpulso negativo, un resultado que dado el alto rango de variación de la consigna y las limitadas características técnicas de esta tarjeta son justificadas. En la Figura 77 b) se presenta la señal correspondiente a la acción que el controlador ejerce en el actuador de la planta, en la cual se aprecia un comportamiento similar al de la variable de proceso.



a)



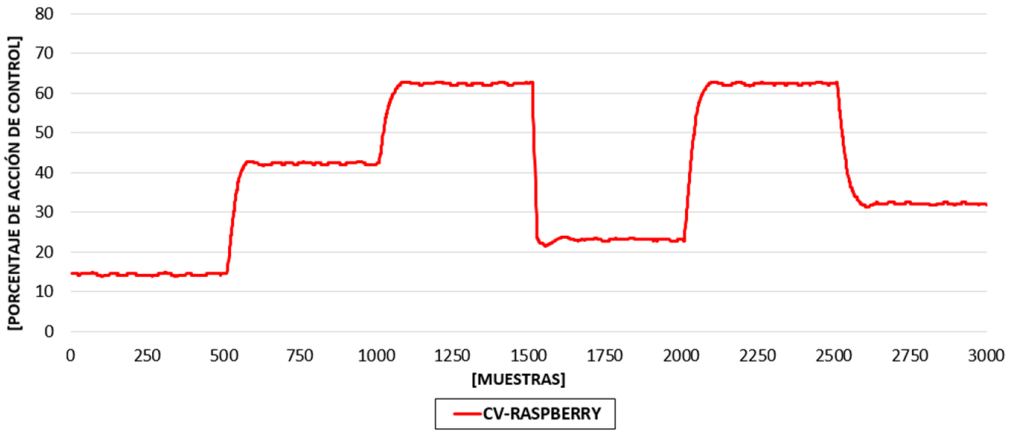
b)

Figura 77. a) Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar el algoritmo de control borroso en la tarjeta Arduino UNO R3.

5.1.2. Pruebas en la tarjeta Raspberry Pi 3

En la Figura 78 a) y b) de forma similar se muestra el comportamiento de las variables de proceso y de control a los mismos cambios de consigna, cuando el algoritmo es implementado en Raspberry Pi. En la parte a) se puede evidenciar como a lo largo de las pruebas la respuesta de PV es rápida, inclusive cuando el cambio de SP es abrupto, el sobreimpulso que observa es mínimo, lo cual concuerda con lo esperado, ya que dicha tarjeta posee la mayor capacidad de procesamiento con respectos a las demás.

a)

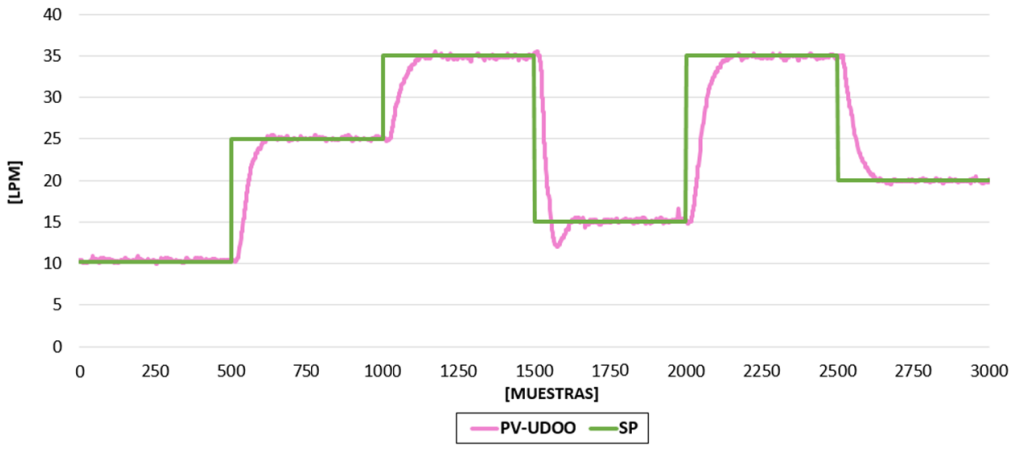


b)

Figura 78. a) Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar el algoritmo de control borroso en la tarjeta Raspberry Pi 3.

5.1.3. Pruebas en la tarjeta Udoo Neo Full

En la Figura 79 a) y b) se presentan los resultados al realizar la misma prueba, pero ahora implementando el algoritmo en la tarjeta Udoo Neo Full. Como se mencionó en el capítulo anterior se utilizó el procesador Cortex-A9 (interfaz similar a Arduino), donde la variable de proceso y de control presentaron un comportamiento similar al que se observa al utilizar la tarjeta Arduino.



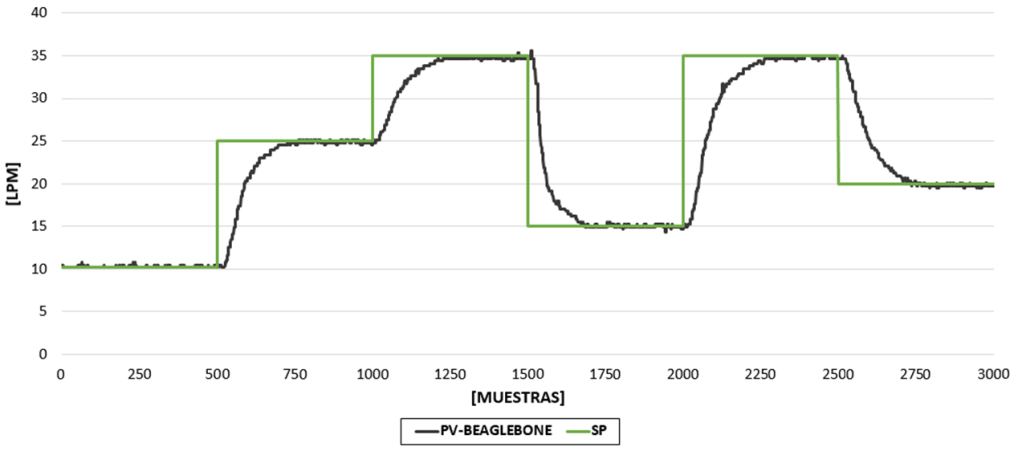
a)

b)

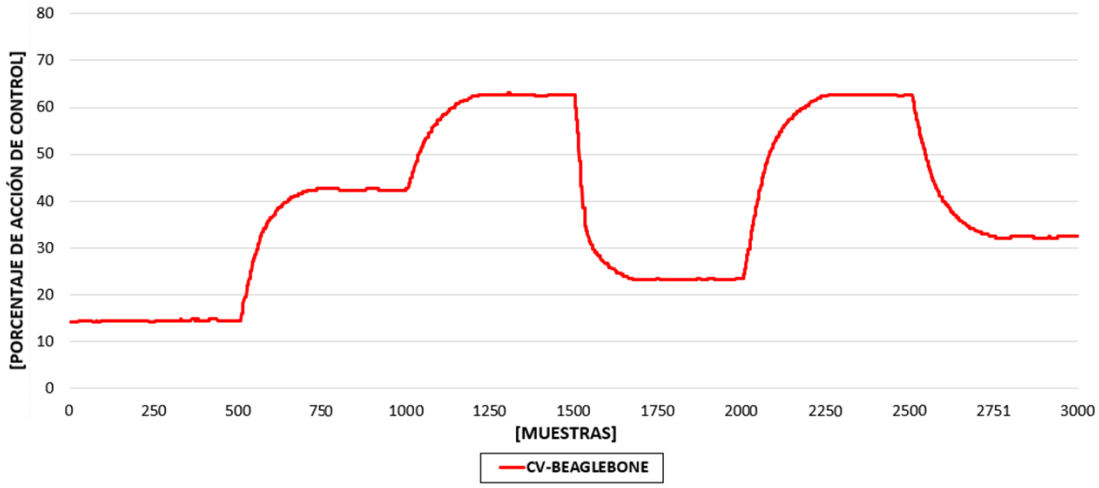
Figura 79. a) Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar el algoritmo de control borroso en la tarjeta Udoo Neo Full.

5.1.4. Pruebas en la tarjeta Beaglebone Black

Al utilizar Beaglebone Black se observa que el comportamiento de las curvas de las señales provenientes de PV y CV reaccionan de una forma más lenta que con las tarjetas antes evaluadas, ya que su procesador opera a una menor capacidad, como se muestra en la Figura 80 a) y b).



a)

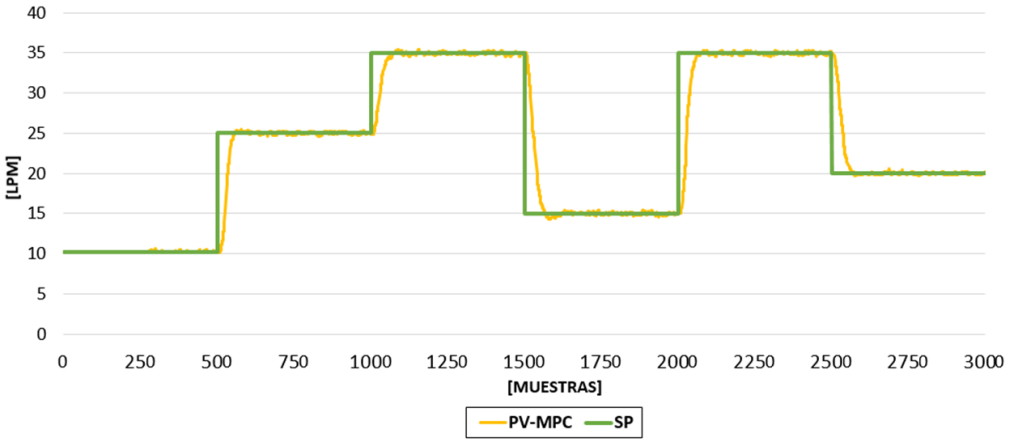


b)

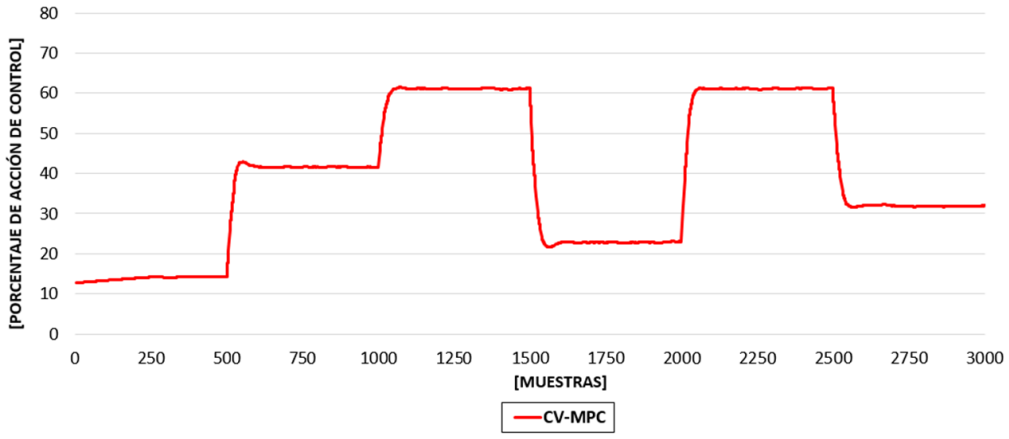
Figura 80. a) Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar el algoritmo de control borroso en la tarjeta Beaglebone Black

5.2. Análisis del controlador MPC

Para evaluar los resultados del segundo controlador avanzado se han utilizado los mismos cambios de SP en el mismo intervalo de tiempo que se explicó anteriormente y dichos resultados son presentados en la Figura 81. Como se puede notar la respuesta del proceso al cambio de SP es bastante rápida y la señal de control no presenta oscilaciones.



a)

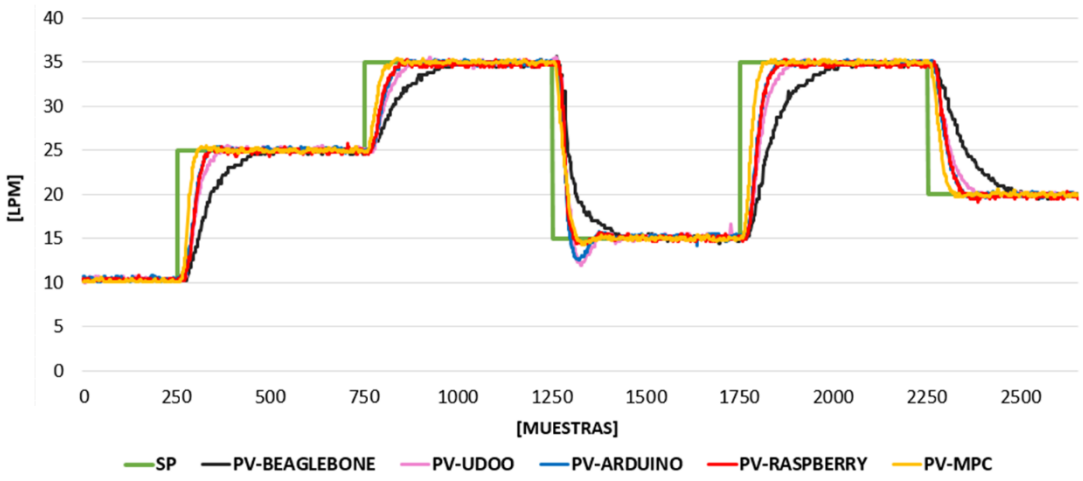


b)

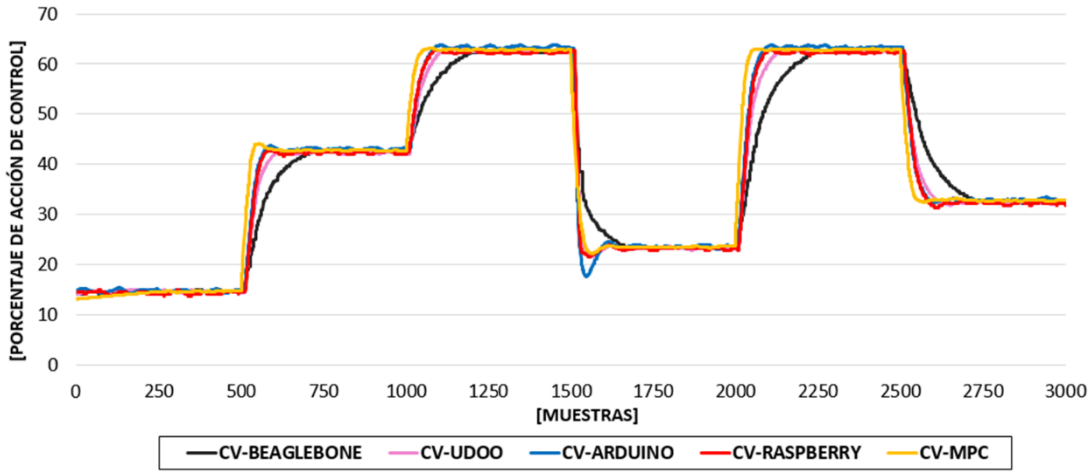
Figura 81. Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar el algoritmo de control predictivo por modelo.

5.3. Comparación entre los controladores borroso y MPC

Luego de haber visto el comportamiento individual del controlador borroso implementado en las cuatro tarjetas y el controlador MPC en Raspberry Pi, se presenta en la Figura 82, las señales de PV y CV en una misma gráfica para una mejor visualización y concepción del rendimiento de cada una.



a)

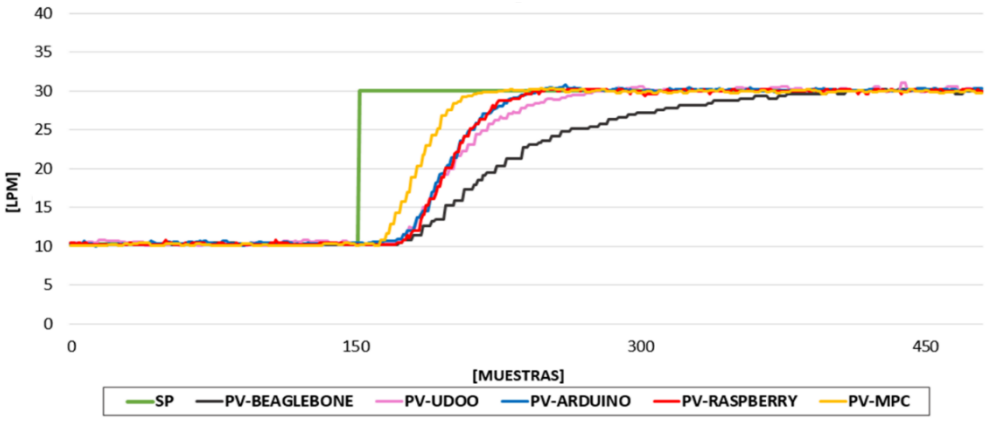


b)

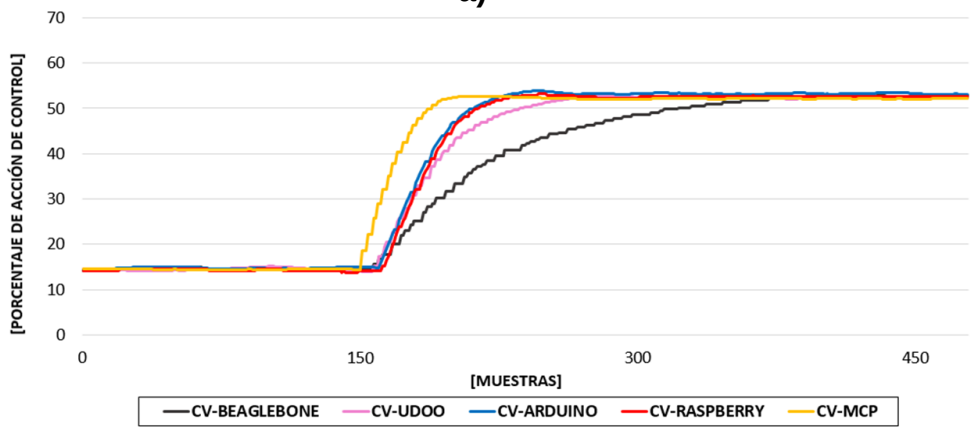
Figura 82. a) Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar los controladores avanzados con distintos valores de SP.

A pesar de que ambos controladores presentan resultados de alta eficiencia, como se puede apreciar la respuesta del controlador MPC es más rápida y con menos sobreimpulso que la conseguida con el controlador borroso, que dependiendo de la tarjeta en la cual se ha implementado varía en sus resultados.

A continuación, se presentan los valores característicos temporales y de sobreoscilación del sistema ante la excitación de un escalón unitario que se produce al cambiar el valor de la consigna de 10 a 30 [LPM], los resultados se muestran en la Figura 83.



a)



b)

Figura 83. a) Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar los controladores avanzados cuando el sistema es excitado con un escalón unitario

El rango de variación del SP es igual a 20 [LPM], por lo tanto, el tiempo de asentamiento (t_s) se determina cuando la curva de respuesta alcanza un rango igual o menor al 5% alrededor del valor que se desea. En este caso dicho rango será dado por $20 \pm 5\%$, es decir entre 29 y 31 [LPM]. El tiempo de subida se determina desde que la curva de respuesta está al 10% hasta que alcanza el 90%, es decir desde 12 hasta 28 [LPM]. En la tabla 11 se presentan dichos parámetros.

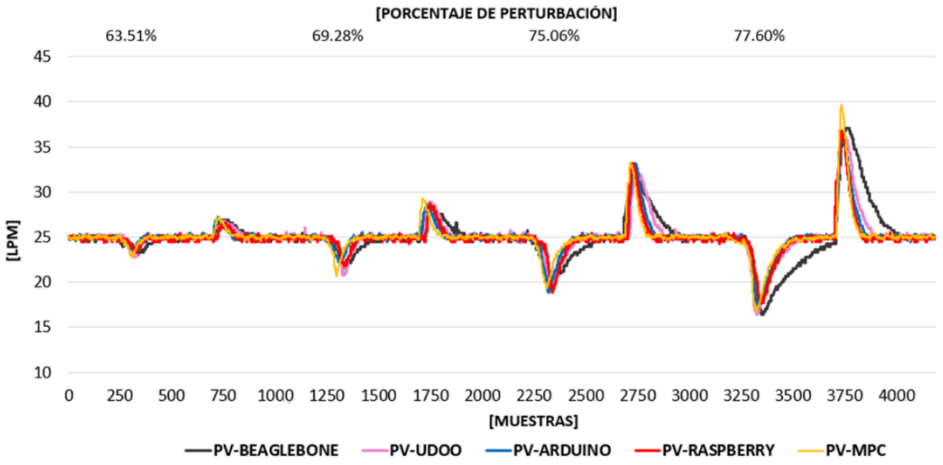
Tabla 11.

Valores característicos temporales y de sobreoscilación que presenta cada una de las tarjetas ante un escalón

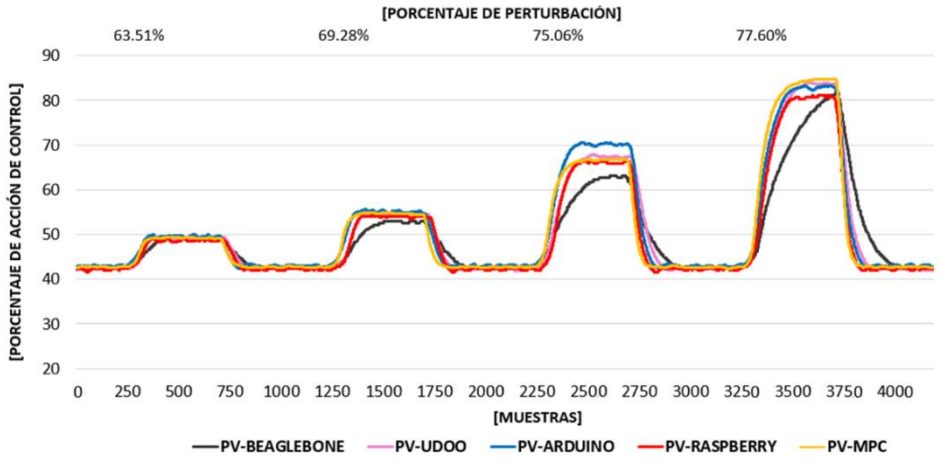
		Tarjeta				
		Arduino	**Raspberry**	**Beaglebone**	**Udoo**	**MPC**
Valores Característicos y de Sobreimpulso	**Tm**	0.8s	0.76s	0.88s	0.8s	0.44s
	Tr	1.92s	1.8s	4.12s	2.49s	1.36s
	Ts	3.16s	3.2s	7.11s	4.24s	2.16s
	%M	3%	0.25%	0.25%	2%	0.75%

Finalmente, se evalúa el comportamiento de los controladores cuando existe una perturbación en el proceso, dichos porcentajes de perturbación seleccionados son: 63.51%, 69.28%, 75.06% y 77.06%; tomando en cuenta que si la perturbación es mayor a 77.06% a pesar que la acción de control sea del 100%, la variable de proceso no alcanza el valor esperado, lo cual podría provocar daños en la estación. En la Figura 84, se muestran las curvas que se obtuvieron realizando la misma prueba a todas las tarjetas por igual.

Al realizar un análisis global de los datos obtenidos, se evidencia que, dentro de la implementación del controlador borroso, Raspberry Pi 3 es quien mejores resultados ha presentado, con un tiempo de establecimiento igual a 2.49s ante una excitación de escalón unitario. Siguiendo con lo esperado, el controlador MPC brindó aún mejores resultados, permitiendo que la señal de la variable de control no presente oscilaciones, un factor que busca aumentar la vida útil del actuador.



a)



b)

Figura 84. a) Curvas de respuesta de SP vs. PV. b) Curva de respuesta de la variable CV al implementar los controladores avanzados cuando el sistema es sometido a varios porcentajes de perturbación

CAPÍTULO VI

6. CONCLUSIONES Y RECOMENDACIONES

6.1. Conclusiones

- La variedad de tarjetas embebidas que se encuentra en el mercado crece exponencialmente, no solo en marcas, sino en las versiones de las ya existentes, dependiendo de las necesidades y economía del usuario; para este trabajo se eligieron versiones potentes de microcomputadoras como lo son: Raspberry Pi 3, Beaglebone Black, Udoo Neo Full y la versión más básica e icónica de Arduino (UNO), las cuales ofrecieron buenas prestaciones en relación a costos reducidos.

- Se ha corroborado que las denominadas tecnologías de bajo costo a pesar de sus limitaciones técnicas permiten la implementación de control avanzado, mediante la utilización de programación de alto nivel, librerías especializadas y toolboxes de softwares matemáticos desarrollados específicamente para este tipo de dispositivos.

- El diseño del algoritmo de control borroso con inferencia tipo Mamdani se realiza con relativa facilidad, ya que no requiere el modelo matemático del proceso que se va a controlar; a pesar de ello ha presentado resultados bastante eficientes y dado que los requerimientos computaciones no son tan altos, ha permitido que se lo implemente en todas las tarjetas antes mencionadas, teniendo como base un código muy similar en todos los casos.

- La implementación del controlador MPC en tarjetas embebidas de bajo costo se realiza con mayor facilidad mediante la herramienta Simulink, siempre y cuando disponga de los paquetes de soporte adecuados para trabajar en ellas; el paquete para Raspberry Pi 3 incluye bloques para leer y escribir datos mediante comunicación I^2C, para Beaglebone Black se carece de ellos.

- El control MPC solo se pudo implementar en la tarjeta Raspberry Pi 3, por las prestaciones de la herramienta Simulink; en Arduino UNO R3 no se logró realizarlo debido a que el controlador genera un coste computacional alto y la capacidad de cálculo de esta tarjeta no es suficiente; en Beaglebone Black y Udoo Neo Full a pesar de tener capacidades de cálculo suficientes, no se dispone de los complementos necesarios y por ello tampoco se lo realizó.

- La tecnología que presentó mejores resultados ante la implementación de un controlador borroso y la única que permitió el desarrollo de un controlador MPC fue Raspberry Pi 3, a pesar de solo tener disponibilidad para señales de entrada y salida digitales, se demostró que utilizando elementos externos y acondicionando dichas señales se pueden desarrollar aplicaciones de control que ofrezcan resultados competitivos a los que brindan dispositivos de control industrial.

6.2. Recomendaciones

- El código desarrollado para el controlador borroso a pesar de la similitud en todas las tarjetas, presenta variaciones en la forma en cómo se realiza la comunicación serial, por lo cual es adecuado verificar que buses de comunicación están activos y cuales están disponibles para establecer dicha comunicación.

- Cuando se envíe tramas por comunicación serial es recomendable usar programas externos (por ejemplo, el programa X-CTU), para verificar que la información que llega al puerto sea la adecuada, esto puede ayudar a encontrar y corregir rápidamente errores que se comenten al enviar datos por comunicación serial.

- Udoo en todas sus versiones presenta en su entorno una interfaz similar a la de Arduino en su escritorio remoto, en el cual se pueden cargar programas desarrollados en Arduino IDE, con la única diferencia que al momento de escribir en el puerto serial se debe utilizar el comando Serial0.

- Para implementar el controlador MPC en la tarjeta Raspberry Pi 3, es recomendable instalar la versión 2017a de MATLAB, esto se debe a que solo en esta versión Simulink posee los bloques de comunicación I2C y SPI.

REFERENCIAS BIBLIOGRÁFICAS

Abril, E., & Pacheco, A. (2012). Análisis, diseño e implementación de un control de velocidad difuso aplicado al robot móvil robotino. Quito: Universidad Politécnica Salesiana Sede-Quito Sur. Recuperado el 25 de Junio de 2017

Aftab, M., Chen, C., Chau, C.-K., & Rahwan, T. (2017). Automatic HVAC Control with Real-time Occupancy Recognition and Simulation-guided MoChi-Kindel Predictive Control in Low-cost Embedded System. Energy and Buildings - Journal - Elsevier, 1-17. Recuperado el 29 de Mayo de 2017

Aguilar, L. M., & Ortiz, C. (2017). Análisis de la tecnología ardupilot orientada a la implementación de sistemas de control automático. Cuenca: Universidad Politécnica Salesiana Sede Cuenca. Recuperado el 12 de Junio de 2017

Blasco Ferragud, F. X. (s.f.). Control predictivo basado en modelos mediante técnicas de optimización heurística. Aplicación a procesos no lineales y multivariables. Valenecia: Universidad Politécnica de Valenia. Recuperado el 20 de Septiembre de 2017

Carvajal, C., & Proaño, L. (2015). Diseño e implementación de una estación de caudal y puesta en servicio de un transmisor magnético de flujo, para el monitoreo y control automático de la variable caudal, en el laboratorio de redes industriales y control de procesos de la ESPE. Latacunga. Recuperado el 09 de Diciembre de 2017

Castiñeira, H. (s.f.). Educación Tecnológica. Recuperado el 14 de Agosto de 2017, de http://www.tecnologia-tecnica.com.ar/sistemadecontrol/index%20sistemasdecontrol.htm

Chabni, F., Taleb, R., Bembouali, A., & Bouthiba, M. A. (2016). The Application of Fuzzy Control in Water Tank Level Using Arduino. (IJACSA) International Journal of Advanced Computer Science and Applications, 7(4), 261-265. Recuperado el 2017 de Mayo de 20, de http://ahmetcevahircinar.com.tr/wp-content/uploads/2016/11/The_Application_of_Fuzzy_Control_in_Water_Tank_Level_Using_Arduino.pdf

Chacón Galarza, G., & Tapia Tapia, V. (2017). Sistema de monitoreo y control de la temperatura de flujo de aire mediante hardware y software libre para su uso didáctico en el aprendizaje de control automático. Latacunga. Recuperado el 20 de Septiembre de 2017

Cisneros, J. P. (2016). Diseño de un controlador inteligente usando técnicas de lógica difusa, aplicado a un balastro electrónico. Aguascalientes: Universidad Autónoma de Aguascalientes. Recuperado el 20 de Agosto de 2017

Debian. (09 de Junio de 2017). Debian. Recuperado el 09 de Diciembre de 2017, de https://www.debian.org/intro/about

Dulhoste, J.-F. (s.f.). Teoría de Control. Recuperado el 15 de Septiembre de 2017, de http://webdelprofesor.ula.ve/ingenieria/djean/index_archivos/Documentos/TC1_IntroduccionSC.pdf

EcuRed. (10 de Mayo de 2017). EcuRed. Recuperado el 09 de Dicicembre de 2017, de https://www.ecured.cu/MATLAB

E-ducativa. (09 de Marzo de 2017). E-ducativa. Recuperado el 11 de Noviembre de 2017, de http://e-ducativa.catedu.es/44700165/aula/archivos/repositorio/4750/4926/html/15_controlador_de_accin_proporcional_integral_y_derivativa_pid.html

Fernández, M. (07 de Agosto de 2006). El ABC de la automatización. Recuperado el 22 de 05 de 2017

Feroldi, D. (20 de Septiembre de 2012). Control predictivo basado en modelo con restricciones. Recuperado el 15 de Junio de 2017, de http://handbook.usfx.bo/nueva/vicerrectorado/citas/TECNOLOGICAS_20/Electronica/9.pdf

Hentzelt, S., Klingler, A., & Graichen, K. (2014). Experimental results for distributed model predictive control applied to. 2014 IEEE International Symposium on Intelligent Control (ISIC) (págs. 1100-1106). Antibes. doi:10.1109/ISIC.2014.6967614

Herrera Aristizábal, S., & Hincapié Correa, J. (2016). Implementación de algoritmos de control de velocidad y torque de motores de corriente continua aplicados a un modelo de AGV multipropósito de potencia

media utilizando Raspberry Pi. Pereira: UNIVERSIDAD TECNOLÓGICA DE PEREIRA. Recuperado el 15 de Mayo de 2017

HUBOR. (15 de Agosto de 2015). HUBOR. Recuperado el 09 de Diciembre de 2017, de http://www.hubor-proteus.com/proteus-pcb/proteus-pcb/2-proteus.html

Kepware. (16 de Agosto de 2017). KEPServer EX. Recuperado el 09 de Diciembre de 2017, de https://www.kepware.com/en-us/products/kepserverex/

Kurniawan, A. (2015). Arduino Uno: A Hands-On Guide for Beginner. Barcelona: PE Press. Recuperado el 26 de Septiembre de 2017

Kurniawan, A. (2015). BeagleBone Black Programming using Matlab. PE Press. Recuperado el 25 de Junio de 2017

Kurniawan, A. (2016). Getting Started with Raspberry Pi 3. PE Press. Recuperado el 26 de Agosto de 2017

Limón Marruedo, D. (2002). Control predictivo de sistemas no lineales con restricciones: estabilidad y robustez. Sevilla: Universidad de Sevilla. Recuperado el 26 de Julio de 2017

Linux. (19 de Agosto de 2017). eLinux. Recuperado el 1 de Diciembre de 2017, de https://elinux.org/Beagleboard:BeagleBoneBlack#Hardware_Files

MathWorks, Inc. (2015). Simulink and Raspberry Pi Workshop Manual. USA. Recuperado el 09 de Diciembre de 2017

MathWorks, Inc. (15 de Febrero de 2017). Model Predictive Control Toolbox. Recuperado el 09 de Diciembre de 2017, de https://es.mathworks.com/products/mpc.html

Matlab. (2015). Apuntes de Simulink: IQ753 Diseño de Reactores Químicos. Recuperado el 15 de 08 de 2017

Mazzone, V. (2002). Controladores PID. Quilmes. Recuperado el 11 de Noviembre de 2017, de http://www.eng.newcastle.edu.au/~jhb519/teaching/caut1/Apuntes/PID.pdf

Miranda Medrano, J. A. (2017). Fundamentos De Medición Y Control De Procesos. Palibrio. Recuperado el 16 de Mayo de 2017

National Instruments. (12 de Febrero de 2015). National Instruments. Recuperado el 09 de Diciembre de 2017, de http://www.ni.com/white-paper/14556/es/

NCS, C. R. (s.f.). Control Robusto Aplicado a NCS. Recuperado el 12 de Noviembre de 2017, de http://bibing.us.es/proyectos/abreproy/70324/fichero/2.Capitulo+2+-+Control+Robusto+Aplicado+a+NCS.pdf

Núñez Enríquez, F. (30 de Agosto de 2007). UDLAP: Bibliotecas. Recuperado el 10 de Noviembre de 2017, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lep/nunez_e_f/capitulo1.pdf

Piedrafita Moreno, R. (Diciembre de 1999). UNIZAR. Recuperado el 25 de Agosto de 2017, de http://automata.cps.unizar.es/regulacionautomatica/historia.PDF

Prof. Castillo Rubio, P. (29 de Marzo de 2008). SlideShare. Recuperado el 10 de Noviembre de 2017, de https://www.facebook.com/RONNYJAGUARSIMPSON/videos/vb.1962540860645323/2062546567311418/?type=2&theater

Quirarte, A. (05 de Junio de 2014). Hacedores. Recuperado el 11 de Septiembre de 2017, de ¿Qué tarjeta de desarrollo elegir?: http://hacedores.com/que-tarjeta-de-desarrollo-elegir-parte-1/

Ramirez Ramos, O. (12 de Junio de 2008). UDLAP: Bibliotecas. Recuperado el 11 de Noviembre de 2017, de http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/ramirez_r_o/capitulo_2.html

Redondo Fonseca, M. (2016). Simulación de redes neuronales como herramienta Big Data en el ámbito sanitario. Lulu.com. Recuperado el 29 de Agosto de 2017

RS-COMPONENTS. (s.f.). DOCS-EUROPE. Obtenido de http://docs-europe.electrocomponents.com/webdocs/14ba/0900766b814ba5fd.pdf

Ruiz Canales, A., & Miguel, J. (2010). Automatización y telecontrol de sistemas de riego. Barcelona: Marcombo. Recuperado el 16 de Mayo de 2017

TEXAS INSTRUMENTS. (14 de Mayo de 2014). TEXAS INSTRUMENTS. Recuperado el 28 de Septiembre de 2017, de http://www.ti.com/tool/BEAGLEBK

Thaler, G. J. (1974). Automatic control: Classical Linear Theory. Pennsylvania: Stroudsburg. Recuperado el 22 de Mayo de 2017

Torres, D. H. (01 de Enero de 2015). HETPRO. Recuperado el 09 de Diciembre de 2017, de https://hetpro-store.com/TUTORIALES/beaglebone-black-introduccion/

Ubuntu. (21 de Enero de 2017). Ubuntu MATE. Recuperado el 09 de Diciembre de 2017, de https://ubuntu-mate.org/raspberry-pi/

UDOO. (25 de Mayo de 2016). UDOO. Recuperado el 1 de Diciembre de 2017, de https://www.udoo.org/udoo-neo/

UDOO. (08 de Octubre de 2016). UDOO Neo. Recuperado el 01 de Diciembre de 2017, de https://www.udoo.org/docs-neo/Hardware_Reference/Board%20versions.html

Vaca, A. E., & Curay, S. D. (Enero de 2015). Diseño e implementación de un algoritmo de control avanzado aplicado a un proceso de presión, utilizando un controlador de automatización programable para el laboratorio de redes industriales y control de procesos de la Universidad de las Fuerzas Armadas. Latacunga, Cotopaxi, Ecuador: Universidad de las Fuerzas Armadas ESPE Extensión Latacunga. Recuperado el 22 de Mayo de 2017, de http://repositorio.espe.edu.ec/handle/21000/9319

Vargas Herrera, D. (13 de Febrero de 2017). Sistema de Espacios Virtuales para Neuro-rehabilitación. Recuperado el 25 de Agosto de 2017, de http://www.ptolomeo.unam.mx:8080/xmlui/bitstream/handle/132.248.52.100/10619/tesis.pdf?sequence=1

ANEXOS

DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE ELECTRÓNICA E INSTRUMENTACIÓN

CERTIFICACIÓN

Se certifica que el presente trabajo fue desarrollado por los señores: **Jorge Luis Buele León y John Javier Espinoza Mejía.**

En la ciudad de Latacunga, a los 15 días del mes de enero de 2017.

Ing. Marco Pilatásig
DIRECTOR DEL PROYECTO

Aprobado por:

Ing. Franklin Silva Monteros
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO