Facultad de Ingeniería Eléctrica

Departamento de Control Automático

Trabajo de Diploma

Tesis en Opción al Título Académico de Ingeniero en Control Automático.

Título: *"Diseño e implementación de Prácticas de Laboratorio para asignaturas de la carrera de Ingeniería Automática en base al Arduino Uno y MatLab."*

Autor: *Orleans Guilarte González.*

Tutores: *Ing. Jorge Jadid Tamayo Pacheco.*

MSc. Saddid Lamar Carbonell.

"Santiago de Cuba, año 2016"

Pensamientos

“Somos arquitectos de nuestro propio destino”

Albert Einstein.

“Los sueños parecen al principio imposibles, luego improbables, y cuando nos comprometemos se vuelven inevitables”

Mahatma Gandhi.

Dedicatoria

- *A la memoria de mi padre.*
- *A mi mamá, por confiar en mí a pesar de todo y estar presentes en todos los momentos cumbres de mi vida.*
- *A mi hermana querida, por su ayuda ilimitada y sin pretextos, y por ser mi modelo a seguir en estos cinco años de carrera.*
- *A mis primos Robert y Yanelis para que aprendan del maestro.*

Agradecimientos

- *A mi madre por ser la mejor de todas.*
- *A mis tutores Jadid y Saddid, por tantas horas dedicadas a esta causa, por el apoyo y accesoria de forma incondicional. Muchas gracias.*
- *A mi familia y amigos, por todo su apoyo.*
- *A mis compañeros de estudios Leito, Ale, Gola y todos los demás que siempre me brindaron su ayuda desinteresada.*
- *A todos los que de una manera u otra contribuyeron al desarrollo de este trabajo.*

Resumen

En el siguiente trabajo se presenta una plataforma de hardware y software para realizar el desarrollo de Prácticas de Laboratorio de la especialidad de Automática. Para la realización de esta plataforma se utiliza desde el punto de vista de hardware la plataforma Arduino Uno, donde se ha diseñado la electrónica necesaria para que esta sea comunicada con otros procesos. Por otra parte desde el punto de vista de software se trabajó con el MatLab R2012b atendiendo a las características de este para realizar análisis de control .El presente trabajo tiene una gran importancia debido a que tiene como vital objetivo la modernización de las prácticas de laboratorio de las asignaturas de la carrera Ingeniería Automática así como las disciplinas: Integradora, Sistema de Mediciones, Programación, Control, Computación.

Abstract

In this paper, a hardware platform and software is presented for the development of laboratory practice specialty Automatic. For the realization of this platform is used from the point of view of the hardware platform Arduino Uno, which has designed the electronics necessary for this to be communicated with other processes. Moreover from the point of view of software we worked with MatLab R2012b according to the characteristics of this for analysis .Control This work it is very important because it has a vital objective the modernization of laboratory practices the subjects of race and Automatic Engineering disciplines: Integrative, Measurement System, programming, Control, Computer

Índice

INTRODUCCIÓN

La automatización y la electrónica ocupan un lugar importante en la vida actual, puesto que se encuentran presentes en casi todas las áreas de la vida moderna, alcanzando un desarrollo vertiginoso y fortaleciendo de manera significativa las esferas industriales, empresariales y universitarias.

A partir de la segunda década del siglo pasado comienzan a desarrollarse nuevas tecnologías en varios campos de las ingenierías, aparecen los llamados microcontroladores con los cuales se logran equipos eléctricos, electrónicos e informáticos más compactos y versátiles. Algunos ejemplos de los avances de estas tecnologías son: el control automático de una línea de producción determinada, el desarrollo de equipos médicos, los sistemas para el control automático de la red de energía eléctrica, equipos reguladores del factor de potencia, así como el desarrollo de plataformas como la de Arduino, entre otros.

Hoy en día la utilización de la plataforma Arduino se está generalizando en centros de investigación y desarrollo, así como en prestigiosas universidades. Además, pueden ser fabricados infinidad de prototipos y cada vez su uso se viene expandiendo más en los sistemas de automatización, en viviendas (domótica), integración con Internet, equipos analizadores de ADN, entre otros.

Nuestro país no ha estado ajeno a estos avances tecnológicos y realiza arduos esfuerzos para lograr el desarrollo y la implementación de nuevas tecnologías en diferentes áreas del conocimiento. Al realizar un análisis del Modelo del profesional de la carrera de Ingeniería en Automática se nota la importancia del diseño y de la utilización de diferentes plataformas que permitan desarrollar proyectos que complementen las competencias profesionales que adquieren estos ingenieros [11].

La carrera Ingeniería en Automática, perteneciente a las llamadas Ciencias Técnicas, está concebida como una carrera de perfil amplio, que persigue formar a un profesional capaz de desempeñarse en importantes áreas tales como la Instrumentación y el Control de procesos, la Electrónica y la Informática. De ahí que el objetivo formativo general de la carrera, consiste en lograr que los estudiantes

sean capaces de: diseñar y explotar sistemas de medición, de control, electrónicos, e informáticos, utilizando criterios técnico-económicos y ecológicos, las normas de Protección e Higiene del Trabajo, así como las técnicas de dirección y organización de la producción, utilizando adecuadamente la Información Científico Técnica, tanto en idioma español como inglés, haciendo un uso eficiente de la computación y las Tecnologías de la Información y las Comunicaciones, siendo capaces de investigar y ejercer la docencia sobre su profesión.

En la Facultad de Ingeniería Eléctrica de la Universidad de Oriente de Santiago de Cuba se estudia la carrera de Ingeniería en Automática y en el proceso docente educativo de la misma las prácticas de laboratorios juegan un papel fundamental, persiguiendo como principal objetivo el desarrollo de habilidades profesionales en los estudiantes. Ahora bien, no obstante al papel que ocupan las prácticas de laboratorios en el proceso de formación de los profesionales de esta carrera, se ha podido apreciar que mayoritariamente los estudiantes no se sienten suficientemente motivados por la realización de las mismas y en ocasiones, no logran comprender la importancia que tienen en su formación como futuros Ingenieros en Automática. Al respecto, uno de los aspectos que ha incidido en tal situación es la limitada existencia de las mismas, así como su falta de actualidad. Por lo tanto, el desarrollo de nuevas y actuales prácticas de laboratorios se convierte en una vía efectiva para mejorar la motivación de los estudiantes, incidiendo por ende, en el perfeccionamiento de la formación profesional.

En base a lo planteado anteriormente, se define como **problema de la investigación:** Insuficiente empleo de tecnologías modernas en las Prácticas de Laboratorio Docentes de la Especialidad de Automática, específicamente en las disciplinas de Instrumentación y Sistemas de Control.

El **objeto de la investigación** Prácticas de Laboratorio de la especialidad de Automática que responden a las disciplinas de Instrumentación y Sistemas de Control. De ahí que se precise como **campo de acción** la supervisión y control desde la computadora mediante tarjetas microcontroladas.

El **objetivo de la investigación** es el diseño e implementación de Prácticas de Laboratorio Docentes para las disciplinas de Instrumentación y Sistemas de Control, en la Especialidad de Automática, empleando la plataforma Arduino Uno y MatLab.

Como **hipótesis** de la investigación se plantea que si se diseñan e implementan las prácticas de laboratorio utilizando la plataforma de Arduino Uno y MatLab, se podría aumentar y modernizar las prácticas de laboratorios, aumentando la motivación de los estudiantes por las mismas.

Para el cumplimiento del objetivo de la investigación se proponen las siguientes **tareas de investigación:**

1. Caracterizar la plataforma Arduino y el software MatLab para su uso en las prácticas de laboratorio docente de la carrera Ingeniería Automática.
2. Lograr establecer la Interacción Arduino MatLab para el desarrollo de prácticas de laboratorios docentes.
3. Desarrollo de una plataforma de hardware y software para realizar las prácticas de laboratorios docentes.
4. Diseño e Implementación de tres prácticas de laboratorios docentes.

Estas tareas han sido desarrolladas utilizando como base las siguientes **Técnicas y métodos de investigación:**

1. Análisis de documentos. Para realizar la consulta de bibliografía de diferentes autores que trabajan la temática.
2. Método histórico-lógico. Para realizar un análisis histórico sobre la evolución y los avances de la plataforma de Arduino Uno, así como de las Prácticas de Laboratorio Docentes de la Especialidad de Automática.
3. Método de análisis. Para analizar las diferentes alternativas sobre el diseño y la implementación de prácticas de laboratorio utilizando Arduino Uno.
4. Método particular o propio del investigador. Experiencias del autor durante su vida universitaria.

Aporte de la investigación:

El aporte de esta investigación consiste en diseñar una plataforma de hardware y software que pueda ser utilizado en la realización de Prácticas de Laboratorio con el objetivo de apoyar la docencia en la carrera de Ingeniería Automática, así como ampliar y modernizar las mismas.

Significación práctica de la investigación:

Al implementar una cierta cantidad de prácticas de laboratorios nuevas y modernas para la carrera de Ingeniería Automática, utilizando Arduino Uno y MatLab, se logra interactuar con hardware y software más modernos y con mayores prestaciones, así como los estudiantes pueden ir adquiriendo mayores habilidades lo que trae como resultado un aumento de la motivación de estos en cuanto a el proceso de enseñanza-aprendizaje. Además se fomentan las bases para el posible desarrollo de nuevas prácticas de laboratorio.

Organización del Trabajo:

La presente investigación se encuentra organizada con una introducción general, dos capítulos con sus introducciones y conclusiones parciales, y finalmente las conclusiones generales, recomendaciones, bibliografía y anexos correspondientes.

En el Capítulo I se presenta el estudio teórico del presente trabajo. Se hace referencia a la importancia de las prácticas de laboratorio en la enseñanza Automática. También se realiza una descripción de la tarjeta Arduino Uno y se abordan las características del programa MatLab/Simulink que lo hace ventajoso. Además de que se realiza una descripción de la situación actual existente en los laboratorios en el Departamento de Control Automático.

En el Capítulo II se describe el desarrollo de un escudo para la plataforma Arduino y la interacción de este con el software MatLab para la elaboración de tres prácticas de que responden a las disciplinas de Instrumentación y Sistemas de Control.

CAPITULO 1: Laboratorios docentes dentro de la carrera de Ingeniería Automática.

En este capítulo se hace un breve argumento de la importancia, objetivo y consecuencias de las prácticas de laboratorio para la enseñanza. También se realiza la caracterización y la descripción de la plataforma de Arduino Uno y se analizan las ventajas, utilización y aplicación de la misma con fines docentes. Además, se realiza una pequeña explicación de la interacción de la plataforma Arduino Uno con el programa MatLab/Simulink. Por último se realiza una descripción de la situación actual en los laboratorios del Departamento de Control Automático.

1.1 Caracterización histórica de la carrera Automática en Cuba.

Los estudios de automatización tienen sus inicios en Cuba en la década de los años 60 del pasado siglo con la llegada a nuestro país de especialistas checoslovacos en el campo de la automatización. Es a partir del trabajo elaborado por ellos, que comienzan a impartirse asignaturas relacionadas con el Control Automático en las Universidades de Oriente, La Habana y posteriormente en la Universidad de Las Villas.

A finales de la década del 60 se crea el Centro de Investigaciones Digitales en la Universidad de La Habana, cuyo objetivo era el desarrollo de máquinas computadoras. Dado que ya existían maquinas computadoras en el país y era necesario operarlas y darle mantenimiento, surge la necesidad de preparar profesionales para realizar dichas tareas, y es entonces que a partir del curso 70-71 se empieza, en la Universidad de La Habana, a preparar alumnos de quinto año de las especialidades de Control Automático y Telecomunicaciones en un grupo de asignaturas relacionadas con este campo. A partir del curso 90-91 se inicia la preparación de profesionales en la carrera de Ingeniería en Automática y la primera graduación ocurre en el curso 94-95. En el año 1998 se realizan modificaciones a este plan de estudio llegándose a un plan modificado, luego en el año 2007 es aprobado el plan de estudios D tiene su basamento en el criterio original del perfil amplio, pero como tal hace un fuerte hincapié en la integralidad y reduce el tiempo

de presencia del estudiante en el aula con mayor énfasis en el trabajo independiente y la autoformación; este plan es el que continúa aplicándose hasta la actualidad y que se adaptaba a las nuevas necesidades del país.

El modelo del profesional del Ingeniero en Automática recoge los objetivos educativos e instructivos por año académico, así como las diferentes indicaciones metodológicas y las estrategias que deben integrarse durante los años de estudio. Además de que un ingeniero automático debe poseer un gran dominio de las asignaturas básicas como: Matemática, Física, Química, Marxismo - Leninismo, Dibujo e Idioma Inglés, así como una sólida familiarización con los sistemas de medición, control, computación y programación en lenguaje de alto nivel. Lo cual les brinda una base para adquirir conocimientos sobre Sistemas de Control, Instrumentación, Control de Procesos, Sistemas Digitales y Programación entre otras disciplinas relacionadas al Control Automático [17].

1.1.1 Las Prácticas de Laboratorio en la enseñanza

A comienzos del siglo XX, la enseñanza del laboratorio de ciencias tuvo un particular auge con énfasis en los trabajos experimentales, pero entró en conflicto en los años veinte y treinta debido a la importancia que se le comenzó a otorgar a las demostraciones sin evidencias pedagógicas justificables. No obstante, la época del lanzamiento del Sputnik, en 1957, le dio un empuje a la enseñanza de las ciencias en los años sesenta, resurgiendo la enseñanza experimental del laboratorio, ahora con énfasis en el método por descubrimiento.

A pesar de este renacimiento experimental de la enseñanza de la ciencia en los años sesenta, ya para la década del setenta, se observa una declinación en el interés por los laboratorios en general y se comienza a cuestionar su efectividad y objetivos. Parte de este desánimo estaba asociado a los desacuerdos existentes sobre los objetivos del trabajo del laboratorio, poniéndose de manifiesto una situación que no era realmente nueva, ya que desde finales del siglo XIX se había reportado “el caótico trabajo de laboratorio”. No obstante, esta situación de incertidumbre abrió el camino para la investigación sobre su verdadero rol en la enseñanza de las ciencias y los objetivos que persigue.

Hasta finales de los años cincuenta del pasado siglo, la enseñanza del laboratorio se centró principalmente en actividades verificativas discutidas en las clases de teoría, planteadas en los libros de texto o sugeridas en manuales de laboratorio. Esta situación se trató de cambiar con el nuevo curriculum de los años sesenta, dándosele a la enseñanza del laboratorio la función importante de desarrollar habilidades de alto nivel cognitivo, mediante actividades centradas en los procesos de la ciencia a través del método indagatorio.

Un problema general con relación a los objetivos del trabajo de laboratorio detectado en los años sesenta es que los mismos no se correspondían con objetivos propios del trabajo práctico. Esto condujo a que en los años setenta esta situación se tratara de mejorar, pero resultó un fracaso, en virtud de que los objetivos elaborados respondían a los de un curso de ciencia, en general, en el que se enfatizaba el reforzamiento y comprobación de teorías. Hasta mediados de los años noventa, se señalaba que los trabajos de laboratorio tenían como objetivos principales generar motivación, comprobar teorías y desarrollar destrezas cognitivas de alto nivel. Sin embargo, muchos estudiantes piensan que el propósito del trabajo de laboratorio es seguir instrucciones y obtener la respuesta correcta, por lo que se concentran en la idea de manipular instrumentos más que manejar ideas.

La Ciencia es una actividad eminentemente práctica, además de teórica, lo cual hace que en su enseñanza el laboratorio sea un elemento indispensable. Sin embargo, a pesar de su papel relevante para el estudio de las ciencias, en la realidad apenas se pueden realizar buenas prácticas de laboratorio en nuestros centros debido a la escasez de recursos y de material de laboratorio. Además de la consideración tradicional de la enseñanza de las ciencias, basada en la transmisión de conocimientos ya elaborados.

El objetivo fundamental de los trabajos en los laboratorios es fomentar una enseñanza más activa, participativa e individualizada, donde se impulse el método científico y el espíritu crítico. De este modo se favorece que el alumno: desarrolle habilidades, aprenda técnicas elementales y se familiarice con el manejo de instrumentos y equipos. Por otra parte, el enfoque que se va a dar a los trabajos

prácticos va a depender de los objetivos particulares que se deseen conseguir tras su realización. Además, las prácticas de laboratorio en la formación de un ingeniero formado en esta especialidad de Automática, es muy importante ya que con las prácticas es donde este puede ejercitar los conocimientos aprendidos teóricamente, donde puede emplear estos conocimientos además de ejercitarlos y les brinda habilidades para poder desarrollarse una vez egresados, en la industria, dándoles una mejor visión de cómo resolver los problemas a los que este se enfrente.

La realización de las prácticas permite poner en desarrollo el pensamiento espontáneo del alumno, al aumentar la motivación y la comprensión respecto de los conceptos y procedimientos científicos. Esta organización permite la posibilidad de relacionarse continuamente entre ellos, y con el profesor. Para que esto funcione adecuadamente, es aconsejable conocer bien su planteamiento, y mediante el uso de la imaginación y de este conocimiento, intentar sacar partido de la, en la mayoría de los casos, deficiente dotación de material de laboratorio con el que se cuenta. Donde a pesar del déficit de instrumentos, materiales de trabajo, equipos, etc… se hace necesario el empleo de tecnologías novedosas.

De esta forma se hace muy importante implementar, mediante el uso de nuevas tecnologías, mejoras al material didáctico existente para la carrera de Ingeniería Automática, con el fin de elevar el rendimiento y nivel en los conocimientos de los estudiantes. Fortalecer el proceso enseñanza-aprendizaje en las prácticas de laboratorio dentro de las disciplinas relacionadas al Control Automático. Lo que conlleva a que los estudiantes tengan una formación científico-tecnológica acorde con las demandas profesionales actuales. También crear en el estudiante la inquietud de desarrollar por su cuenta el uso de las nuevas tecnologías, que le permitan un mejor desempeño en su vida académica y profesional. Además de poder intercambiar experiencias con personal académico de otras instituciones de educación superior, con el objeto de mejorar el contenido de las asignaturas en las disciplinas de la rama, tanto en el aspecto teórico como experimental.

Por otra parte la verdadera intención es conseguir que el laboratorio responda con sus prácticas al ámbito de conocimiento que cubren las disciplinas asignadas a este

departamento en el actual plan de estudios, ampliando el abanico de prácticas disponibles para los alumnos. Esto podría lograrse a un coste económico reducido gracias a la utilización de la arquitectura electrónica digital de la plataforma Arduino (controlador) y a dispositivos electrónicos (sensores y actuadores) básicos. Así mismo, en muchas de las prácticas comerciales en la actualidad, la adquisición de datos físicos opera a través de dispositivos que el alumno percibe con más facilidad. Finalizando, la importancia del uso de tecnologías modernas en los laboratorios se refleja en los estudiantes de modo que:

El alumno adquirirá:

- Habilidad para trabajar en grupo.
- Capacidad de análisis y síntesis.
- Habilidad y métodos para la resolución de problemas.
- Capacidad de organización y planificación.
- Razonamiento crítico.
- Creatividad.
- Capacidad para aplicar la teoría a la práctica.
- Capacidad de modelar un problema científico e implementar una solución práctica a un problema.

El alumno será capaz de:

- Entender los fenómenos físicos estudiados teóricamente.
- Analizar e interpretar de datos experimentales.
- Evaluar resultados y compararlos con las predicciones de la teoría.
- Familiarizarse con las técnicas de control automático y adquisición de datos.
- Enfrentarse a fenómenos científico-técnicos concretos, aplicando los conocimientos adquiridos.
- Resolver problemas y aprovechar oportunidades, aprendiendo a diferenciar ambos.
- Adquirir destreza en el uso de sistemas básicos de control automático y electrónico y capacidad para mejorar y sugerir cambios en los mismos.

La Ciencia es una actividad eminentemente práctica, además de teórica, lo cual hace que en su enseñanza el laboratorio sea un elemento indispensable. Sin embargo, a pesar de su papel relevante para el estudio de las ciencias, en la realidad apenas se pueden realizar buenas prácticas de laboratorio en nuestros centros debido a la escasez de recursos y de material de laboratorio. Además de la consideración tradicional de la enseñanza de las ciencias, basada en la transmisión de conocimientos ya elaborados.

El objetivo fundamental de los trabajos en los laboratorios es fomentar una enseñanza más activa, participativa e individualizada, donde se impulse el método científico y el espíritu crítico. De este modo se favorece que el alumno: desarrolle habilidades, aprenda técnicas elementales y se familiarice con el manejo de instrumentos y equipos. Por otra parte, el enfoque que se va a dar a los trabajos prácticos va a depender de los objetivos particulares que se deseen conseguir tras su realización. Además, las prácticas de laboratorio en la formación de un ingeniero formado en esta especialidad de Automática, es muy importante ya que con las prácticas es donde este puede ejercitar los conocimientos aprendidos teóricamente, donde puede emplear estos conocimientos además de ejercitarlos y les brinda habilidades para poder desarrollarse una vez egresados, en la industria, dándoles una mejor visión de cómo resolver los problemas a los que este se enfrente.

Este trabajo solo va a tratar los laboratorios en su forma simple, o sea a nivel de maqueta, donde hay una interacción directa del estudiante con la maqueta, con los instrumentos de laboratorios.

1.2 La Tarjeta Arduino

En la actualidad la evolución de los microcontroladores han impulsado en gran medida el desarrollo de la Ingeniería de Control permitiendo diseñar equipos más compactos y versátiles para la medición y control. Es por ello que cosas tan comunes hoy en día como el transporte, las comunicaciones y en general los procesos de producción han tenido una evolución extraordinaria, alcanzando niveles de seguridad y calidad con los que no se contaba algunas décadas atrás.

Con esta evolución de los microcontroladores se han desarrollado nuevas plataformas, una de la más utilizada en el mundo actual es la de Arduino.

1.2.1 Descripción de la plataforma de Arduino.

Arduino es una plataforma de hardware libre, basada en una placa con un microcontrolador y un entorno de desarrollo, diseñada para facilitar el uso de los microcontroladores en proyectos multidisciplinares. El hardware consiste en una placa con un microcontrolador Atmel AVR y puertos de entrada/salida. Los microcontroladores más usados son el ATmega168, ATmega328, ATmega1280, ATmega8 por su sencillez y bajo costo que permiten el desarrollo de múltiples diseños. Por otro lado el software consiste en un entorno de desarrollo que implementa el lenguaje de programación Processing/Wiring y el cargador de arranque (bootloader) que corre en la placa.

Arduino puede tomar información del entorno a través de sus pines de entrada de toda una gama de sensores y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectar a un computadora, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (por ejemplo: Flash, Processing, MaxMSP, C, C++) [4,8,14].

1.2.2 Ventajas del empleo de Arduino.

Hay muchos otros microcontroladores y plataformas con microcontroladores disponibles para la realización de proyectos multidisciplinares. Parallax Basic Stamp, BX-24 de Netmedia, Phidgets, Handyboard del MIT (Massachusetts Institute of Technology), y muchos otros ofrecen funcionalidades similares. Todas estas herramientas organizan el complicado trabajo de programar un microcontrolador en paquetes fáciles de usar. Arduino, además de simplificar el proceso de trabajar con microcontroladores, ofrece algunas ventajas respecto a otros sistemas a profesores y estudiantes:

- Económicas: Las placas de Arduino son más baratas comparadas con otras plataformas de microcontroladores. La versión más cara de un módulo de Arduino tiene un precio que oscila en el mercado entre los 20 USD o 30 USD.
- Multi-Plataforma: El software de Arduino funciona en los sistemas operativos Windows, Macintosh OSX y Linux. La mayoría de los entornos para microcontroladores están limitados a Windows.
- Entorno de programación simple y directa: El entorno de programación de Arduino es fácil de usar para principiantes y lo suficientemente flexible para los usuarios avanzados. Pensando en los profesores, Arduino está basado en el entorno de programación de Processing con lo que el estudiante que aprenda a programar en este entorno se sentirá familiarizado con el entorno de desarrollo Arduino.
- Software ampliable y de código abierto: El software Arduino está publicado bajo una licencia libre y puede ser ampliado por programadores experimentados. El lenguaje puede ampliarse a través de librerías de C++, y si se está interesado en profundizar en los detalles técnicos, se puede dar el salto a la programación en el lenguaje AVR C en el que está basado.
- Hardware ampliable y de Código abierto: En cuanto a hardware los microcontroladores más usados son los ATmega como se nombraron anteriormente. Los planos de los módulos están publicados bajo licencia Creative Commons, por lo que diseñadores de circuitos con experiencia pueden hacer su propia versión del módulo, ampliándolo u optimizándolo. Incluso usuarios relativamente inexpertos pueden construir la versión para placa de desarrollo para entender cómo funciona y ahorrar algo de dinero [7].

1.2.3 Estructura del hardware de Arduino Uno.

Para la realización de este trabajo es utilizado uno de los últimos modelos diseñados y distribuido por la comunidad Arduino. La placa tiene un tamaño de 75x53mm. Su unidad de procesamiento consiste en un microcontrolador ATmega328. Puede ser alimentada mediante USB o alimentación externa y contiene pines tanto analógicos como digitales, lo que permita desarrollar disimiles proyectos de ingeniería. En la **Figura 1. 1** se presenta la forma física del modelo Arduino Uno.

Figura 1. 1: Frontal y reverso de la placa Arduino Uno.

En la **Tabla 1. 1** se muestra un resumen con las principales características de este modelo [12].

Tabla 1. 1: Resumen características Arduino Uno.

Características	**Descripción**
Microcontrolador	ATmega328
Voltaje operativo	5 V
Voltaje de alimentación (límites)	6 - 20 V
Pines digitales E/S	14 (6 proporcionan salidas PWM)
Pines de entrada analógica	6
Memoria Flash	32 kB (ATmega328) de los cuales 0.5 kB son para el bootloader, cargador de arranque.
SRAM	2 kB (ATmega328)
EEPROM	1 kB (ATmega328)
Velocidad del reloj	16 MHz

Dentro de los componentes fundamentales del Arduino se encuentran los **Pines de alimentación (Power Pins)** como se muestra en la la **Figura 1. 2**.

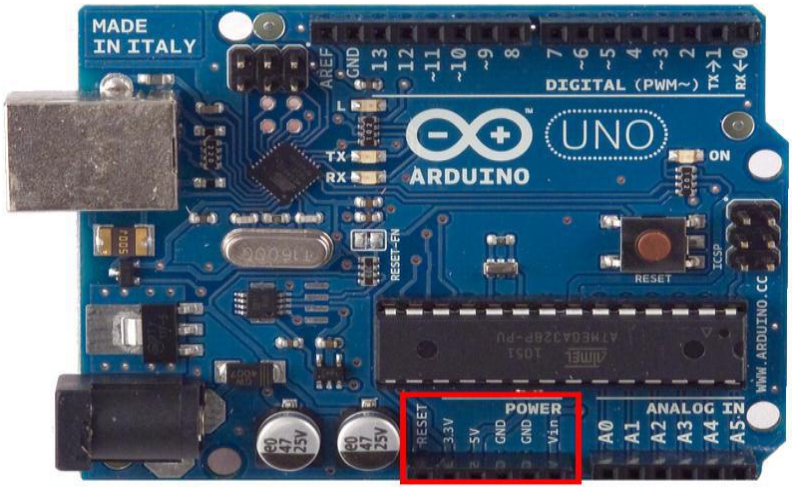


Figura 1. 2: Pines de alimentación (Power Pins).

El Arduino es alimentado mediante la conexión USB o mediante una fuente externa (recomendada de 7-12V), por lo que se obtendrán unas salidas de tensión continua debido a unos reguladores de tensión y condensadores de estabilización.

Estos pines son:

- **VIN**: Es el voltaje de entrada cuando se usa una fuente de alimentación externa (no tiene en cuenta la conexión USB). A través de este pin se puede proporcionar voltaje a la placa, o en caso de estar utilizando una fuente de alimentación externa el pin toma el valor que está siendo suministrado.
- **5V**: Este pin obtiene una tensión de 5 V del regulador de la placa. El regulador es necesario debido a que es alimentado con distintos voltajes. Es una fuente de tensión regulada de 5V, esta tensión puede venir ya sea de pin VIN a través de un regulador interno, o se suministra a través de USB o de otra fuente de 5V regulada.
- **3.3V**: Desde aquí se puede suministrar 3.3 V generados por el regulador interno a los dispositivos que lo necesiten con una corriente máxima de 50 mA.
- **GND**: pines de tierra.
- **Pin de Reset**: Se puede imitar el funcionamiento del botón reset suministrando un valor LOW (0 V) para reiniciar el microcontrolador.

Otros de los componentes fundamentales del Arduino son las **Entradas/Salidas Digitales** se muestran en la **Figura 1. 3**.

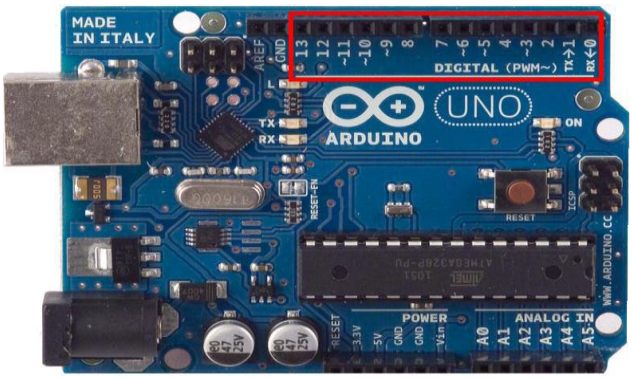


Figura 1. 3: Entradas/Salidas Digitales.

Cada uno de los 14 pines digitales se puede utilizar como una entrada o salida. Cada pin puede proporcionar o recibir un máximo de 40 mA y tiene una resistencia de pull-up (desconectado por defecto) de 20 a 50 kΩ. Además, algunos pines tienen funciones especializadas como:

- Pin 0 (RX) y 1 (TX). Se utiliza para recibir (RX) y la transmisión (TX) de datos serie TTL. Están directamente conectados a los pines serie del microcontrolador. Utilizando estos pines se pueden conectar con otras placas.
- Pin 2 y 3. Interrupciones externas. Se trata de pines encargados de interrumpir el programa secuencial establecido por el usuario.
- Pin 3, 5, 6, 9, 10 y 11. PWM (modulación por ancho de pulso).Constituyen 8 bits de salida PWM (valores de 0 a 255) con la función analogWrite ().
- Pin 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estos pines son de apoyo a la comunicación SPI. Estos pines soportan la librería de comunicación de dispositivos SPI. Integrados para trabajar con información serial sincrónica, proporcionan comunicación SPI, que a pesar de que el hardware la proporcione actualmente no está incluido en el lenguaje Arduino.
- Pin 13. LED. Este pin está conectado con un LED de la placa. Cuando se le asigne un valor HIGH (nivel ALTO) se enciende, en cambio si se deja en LOW (nivel BAJO) estará apagado.

Además de las entradas y salidas digitales el Arduino posee **Entradas Anlògicas** como se muestran en la **Figura 1. 4**

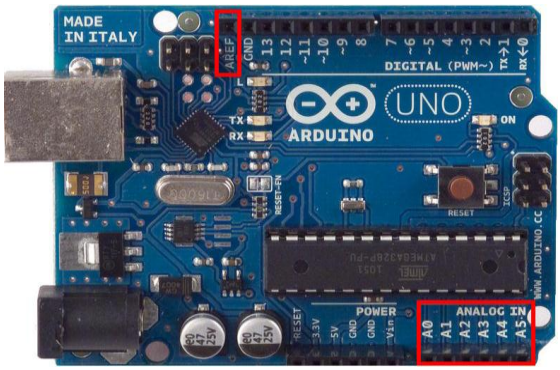


Figura 1. 4: Entradas Analógicas.

El Arduino posee 6 entradas analógicas, etiquetadas desde la A0 a A5, cada una de las cuales ofrecen 10 bits de resolución (es decir, 1024 estados). Por defecto, se tiene una tensión de 5V, pero se podría cambiar este rango utilizando el pin de AREF y utilizando la función analogReference (), donde es introducida una señal externa de continua que es utilizada como referencia.

En la **Figura 1. 5** se muestran otros componentes importantes que se encuentran en la placa según los números señalados en la misma, los cuales son:

1) Conector USB (Universal Serial Bus): Existen varios tipos de conectores USB, en concreto esta placa utiliza el tipo B hembra. Con lo cual se necesitará un cable tipo B macho – tipo A macho (aunque se pueden utilizar otros este es el más extendido) que deberá conectarse a un conector tipo A hembra (por ejemplo a un computadora o al cargador de un móvil). La placa se puede alimentar con la tensión de 5 V que le proporciona el bus serie USB. Cuando se cargue el programa a la placa desde el software de Arduino se inyectará el código de la computadora por este bus.
2) Microcontrolador ATmega328: El microcontrolador es el elemento más importante de la placa. Es donde se instala y ejecuta el código que se haya diseñado. Ha sido creado por la compañía Atmel, tiene un voltaje operativo

de 5 V, aunque se recomienda como entrada de 7-12 V con un límite de 20 V. Contiene 14 pines digitales de entrada y salida, 6 pines analógicos que están conectados directamente a los pines de la placa Arduino comentados anteriormente. Dispone de 32 kB de memoria flash para almacenar código (de los cuales 512 bytes son utilizados por el bootloader, es decir, para el arranque del sistema). Tiene 2 kB de memoria SRAM (Static Random Access Memory) y 1kB de EEPROM (Electrically Erasable Programmable Read-Only Memory).

3) Botón Reset: Utilizando este botón se puede reiniciar la ejecución del código del microcontrolador.
4) Fuente de alimentación externa: La placa puede ser alimentada también mediante corriente directa (CD) suministrada por el conector jack de 3.5mm que podrá recibir entre 7 y 12 V.
5) ICSP (In Circuit Serial Programming): Es un conector utilizado en los dispositivos PIC (Peripheral Interface Controller) para programarlos sin necesidad de tener que retirar el chip del circuito del que forma parte. [4,12]

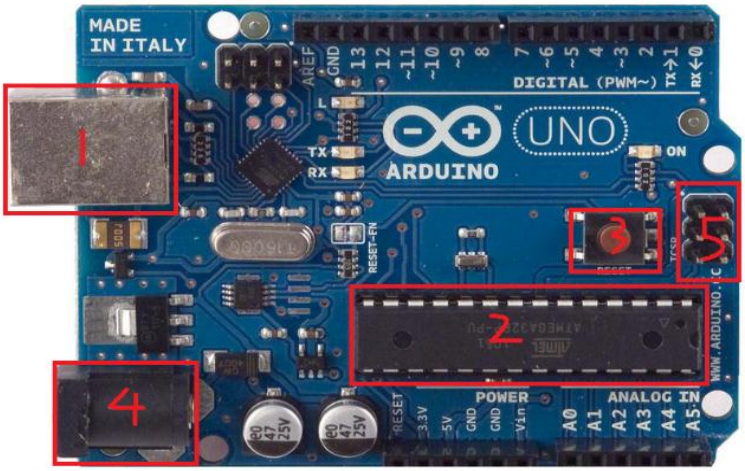


Figura 1. 5: Otros componentes de la placa.

1.2.4 Lenguaje de programación de Arduino.

La plataforma Arduino tiene un lenguaje propio que está basado en C/C++ y por ello soporta las funciones del estándar C y algunas de C++. Sin embargo, es posible utilizar otros lenguajes de programación y aplicaciones populares en Arduino como

Java, Processing, Php, C#, Matlab, Visual Basic, entre otros. Esto es posible debido a que Arduino se comunica mediante la transmisión de datos en formato serie que es algo que la mayoría de los lenguajes anteriormente citados soportan. Para los que no soportan el formato serie de forma nativa, es posible utilizar software intermediario que traduzca los mensajes enviados por ambas partes para permitir una comunicación fluida. Es bastante interesante tener la posibilidad de interactuar con Arduino mediante esta gran variedad de sistemas y lenguajes, ya que dependiendo de cuales sean las necesidades del problema que se va a resolver se aprovecha de la gran compatibilidad de comunicación que ofrece [3,12,14].

1.2.5 Aplicaciones de Arduino.

Las aplicaciones que ofrece Arduino son múltiples, y dependen del problema a resolver. Mediante diferentes componentes se pueden crear aplicaciones sencillas enfocadas a la docencia para estudiantes de la carrera, proyectos más elaborados para la industria, entre otros. Ejemplos de estas aplicaciones:

- Xoscillo: Osciloscopio de código abierto.
- Arduinome: Un dispositivo controlador MIDI (Interfaz Digital de Instrumentos Musicales).
- OBDuino: Un económetro, es un indicador del consumo de un vehículo, que usa una interfaz de diagnóstico a bordo que se halla en los automóviles modernos.
- Humane Reader: Dispositivo electrónico de bajo coste con salida de señal de TV que puede manejar una biblioteca de 5000 títulos en una tarjeta microSD.
- The Humane PC: Equipo que usa un módulo Arduino para emular un computador personal, con un monitor de televisión y un teclado para computadora.
- Ardupilot: Software y hardware de aeronaves no tripuladas.
- ArduinoPhone: Un teléfono móvil construido sobre un módulo Arduino.
- Impresoras 3D [3,4,8,7,9].

1.2.6 MATLAB/SIMULINK y Arduino

A la hora de empezar con cualquier proyecto de aplicación práctica, una de las cuestiones principales suele ser enlazar la parte teórica como algoritmos de tratamiento de datos, sistemas de control automático, etc., con el mundo real. Hasta no hace mucho, la única manera de poder trabajar con datos físicos consistía en adquirir un sistema comercial de adquisición de datos (DAQ) como los de National Instruments; afortunadamente hoy en día hay alternativas más asequibles como la utilización de la plataforma Arduino.

Ventajas del uso del Simulink en Arduino:

La plataforma Arduino ayuda a los estudiantes a comprender el flujo de trabajo para el diseño de un sistema integrado sin necesidad de utilizar la programación manual. Los estudiantes pueden utilizar Simulink para crear algoritmos para sistemas de control y aplicaciones de robótica. Se pueden aplicar técnicas probadas en la industria para el diseño basado en modelos y comprobar que sus algoritmos funcionan durante la simulación. Además de poder implementar los algoritmos en el procesador ATmega en la placa Arduino como aplicaciones independientes. Para poder trabajar con la plataforma Arduino Uno y el MatLab/Simulink es necesario descargar un Tools de Arduino para MatLab donde el Soporte Integrado de Simulink para la plataforma Arduino incluye:

- Instalación automática y configuración
- Biblioteca de bloques de Simulink que se conectan a las E/S de Arduino, tales como digital input y output, analog input y output, serial receive y transmit, y servo read y write.
- Ajuste Interactivo de Parámetros y Monitoreo de Señales en Tiempo real de las aplicaciones que se ejecutan en el Arduino Mega (no disponible en Arduino Uno).
- Implementación de modelos con funcionamiento autónomo.

Soporte de Simulink para Arduino.

El soporte de Simulink para el Arduino hace que el algoritmo que se ejecuta en el microcontrolador contenga una biblioteca de bloques de Simulink (**Anexo 1**), donde se encuentran bloques como Analog Write (Entrada analógica), **figura 1.6** que está a cargo de la medición de tensión de un pin analógico del Arduino. Este bloque de Simulink permite establecer la frecuencia de muestreo para teóricamente un valor tan bajo como 0.000001s, pero tal valor de seguro congelaría la mayoría de las computadoras. Un valor razonable, es un tiempo de muestreo de 0.02s. Valores más bajos que este significarían un cambio en el eje de tiempo escala.

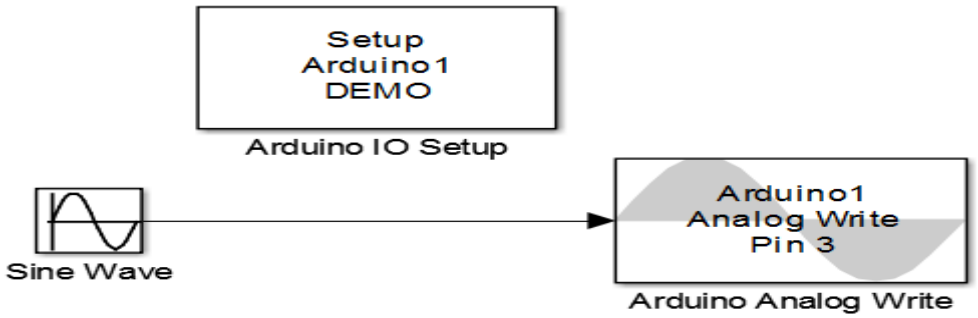


Figura 1. 6: Modelo de Simulink para la prueba de frecuencia.

Otro de los bloques que aporta el soporte Simulink para Arduino es el Analog Read (Salida Analógica), donde el microcontrolador solo contiene un servidor que se ejecuta continuamente donde se puede establecer un valor de tensión en un pin analógico del Arduino, como se muestra en la **figura 1.9**.

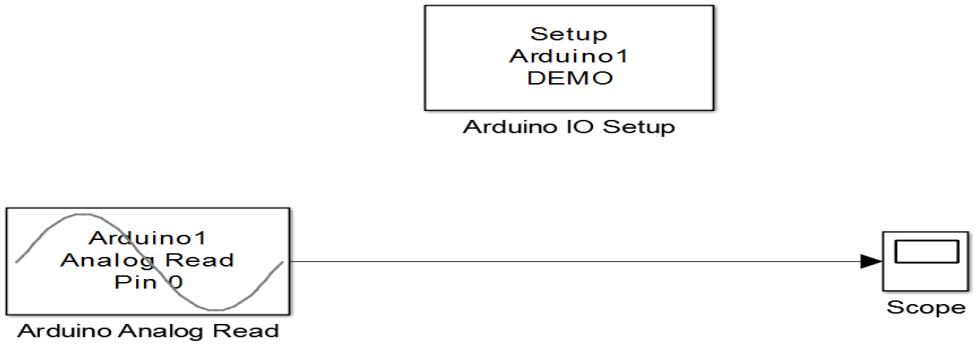


Figura 1. 7: Bloques de Simulink para el paquete Arduino IO.

1.3 Situación de las prácticas de laboratorios en el Departamento de Automática.

A través de conversaciones y entrevistas realizadas a profesores y técnicos que han impartido prácticas de laboratorio durante muchos años se pudieron apreciar cuáles son las distintas situaciones por las que está atravesando el departamento en diferentes disciplinas:

- En la disciplina de Instrumentación se realizan varias prácticas. En las mismas no se emplean tecnologías novedosas, ósea no se realizan prácticas donde haya alguna interacción de las prácticas con alguna plataforma con microcontroladores o se apliquen algunas de estas tecnologías con microcontroladores orientados a la medición o al control de procesos.
- En Sistemas de Medición se utiliza la tarjeta de adquisición de datos PCL-818 que aunque con ella se realizan algunas prácticas de laboratorio presenta desventajas como: que la tarjeta es antigua y su conexión con la PC es a través del bus ISA, bus que solo se encuentra en máquinas antiguas, debido a que las modernas ya no presentan bus y esto obliga a utilizar máquinas antiguas para la realización de las prácticas.
- por lo que si las máquinas son antiguas, los softwares utilizados no pueden ser modernos y esto sería una limitación muy importante. Dentro de estos softwares utilizados uno de los más importantes para la carrera de Ingeniería Automática es el MatLab que sus últimas versiones no pueden ser utilizados con estas tarjetas.
- En el Departamento de Control Automático también se encuentran otras dos tarjetas: la PCL 711b y la super SAD 100 que aunque no se realicen prácticas de laboratorios con estas; son utilizadas para la realización de trabajos de diplomas, trabajos de FORUM, así como otros proyectos. Estas tarjetas al igual que la PCL – 818 tienen como inconveniente que se conectan a la computadora mediante el bus ISA, bus que como ya se encuentra obsoleto en las máquinas modernas representa una gran limitación en el Departamento de Ingeniería Automática para la realización de las prácticas docentes.

Conclusiones Parciales

- El análisis de la caracterización de la carrera de Ingeniería en Automática en Cuba demostró la importancia que tienen las prácticas de laboratorio para los profesionales de esta rama.
- El estudio de los fundamentos teóricos de la plataforma de Arduino Uno permitió valorar las ventajas que presenta la misma para ser utilizado con fines docentes en la realización de prácticas de laboratorio.
- La situación de las prácticas de laboratorio en el Departamento de la carrera de Ingeniería Automática demostró que las funcionalidades y posibilidades de esta plataforma sería de gran avance e importancia en el diseño e implementación de proyectos de ingeniería.

CAPITULO 2: Descripción e Implementación de las Prácticas de Laboratorio.

En este capítulo se desarrolla un escudo de Arduino para logra resolver el inconveniente de las entradas y salidas analógicas de la placa Arduino uno en cuanto al rango de tensión. Además se describen las prácticas de laboratorio docentes diseñadas, las cuales sólo están dirigidas al trabajo con la plataforma Arduino y el software de MatLab R2012b a través del hardware desarrollado y sus respectivas interfaces.

2.1 Desarrollo de escudo de Arduino.

Debido a que la plataforma Arduino uno solo posee entradas aptas para recibir voltajes de 0-5 V y salidas en PWM (modulación de ancho de pulso), el hardware diseñado para la realización de las prácticas laboratorio consta de 3 entradas y 2 salidas analógicas, así como 3 salidas digitales y 2 tierras. Para el acondicionamiento previo de las señales de entrada y salida es necesario la realización de un escudo que monta en la placa Arduino. Aunque hoy en día se puede encontrar en el mercado una gran cantidad de sistemas de adquisición de datos que resuelven las expectativas de las prácticas desarrolladas, el costo es significativamente mayor que el hardware y la aplicación de software descrito en este trabajo.

2.1.1 Diseño de las entradas analógicas 0-10 V

Debido a las especificaciones de diseño del hardware y a que este microcontrolador dispone de seis canales para la conversión A/D (analógico-digital), preparados para manejar voltajes de hasta 5V, es necesario realizar una previa adaptación (**Figura 2. 1**) para trabajar con señales cuyo rango varía entre 0 y 10V. Para conseguir esto se recurre a un divisor de tensión que consta de dos resistencias iguales. En una etapa posterior al divisor se coloca un seguidor de tensión implementado mediante un amplificador operacional (OP) alcanzando con esta configuración la misma tensión a la entrada que a la salida [15].

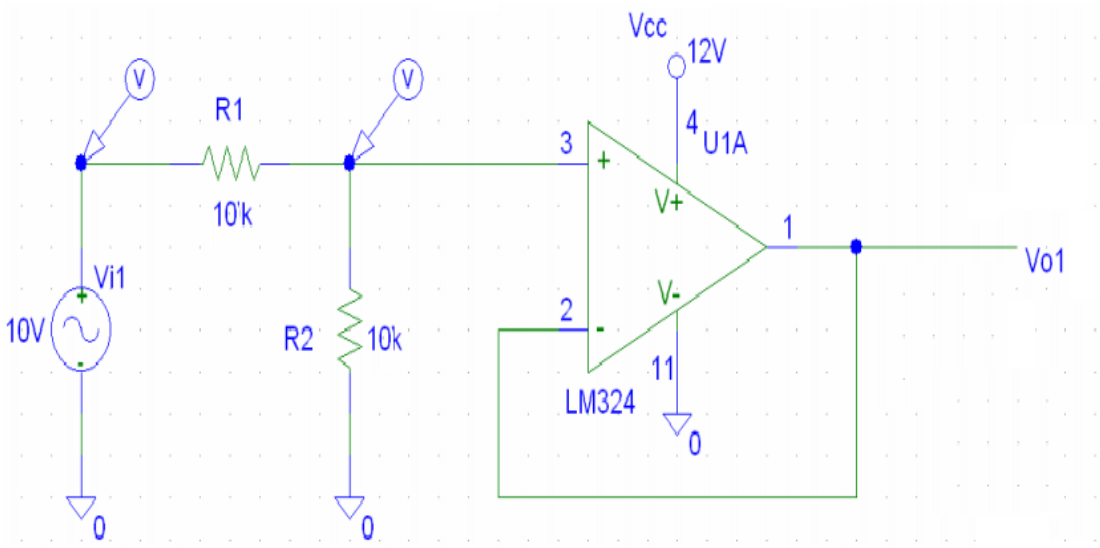


Figura 2. 1: Esquema entrada analógica de 0-10 V.

Para elegir el amplificador operacional se ha optado por el LM324N, un circuito integrado con un amplio rango de alimentación sin necesidad de que sea simétrica o de doble polaridad, lo que facilita su alimentación y en cuyo encapsulado se integran 4 de estos operacionales. Además fueron escogidas las resistencias con un valor nominal de 10KΩ.

En el hardware diseñado el circuito dispone de tres entradas de 0 a 10 V, las cuales se conectan en los pines analógicos A3, A4, A5 respectivamente. Con el objetivo de proteger al microcontrolador de sobre tensiones por efecto de alguna perturbación, se coloca un diodo zener de 5.6V a la salida de los amplificadores.

En la **Figura 2. 2** se muestra el esquema completo de la entrada analógica de 0-10V.

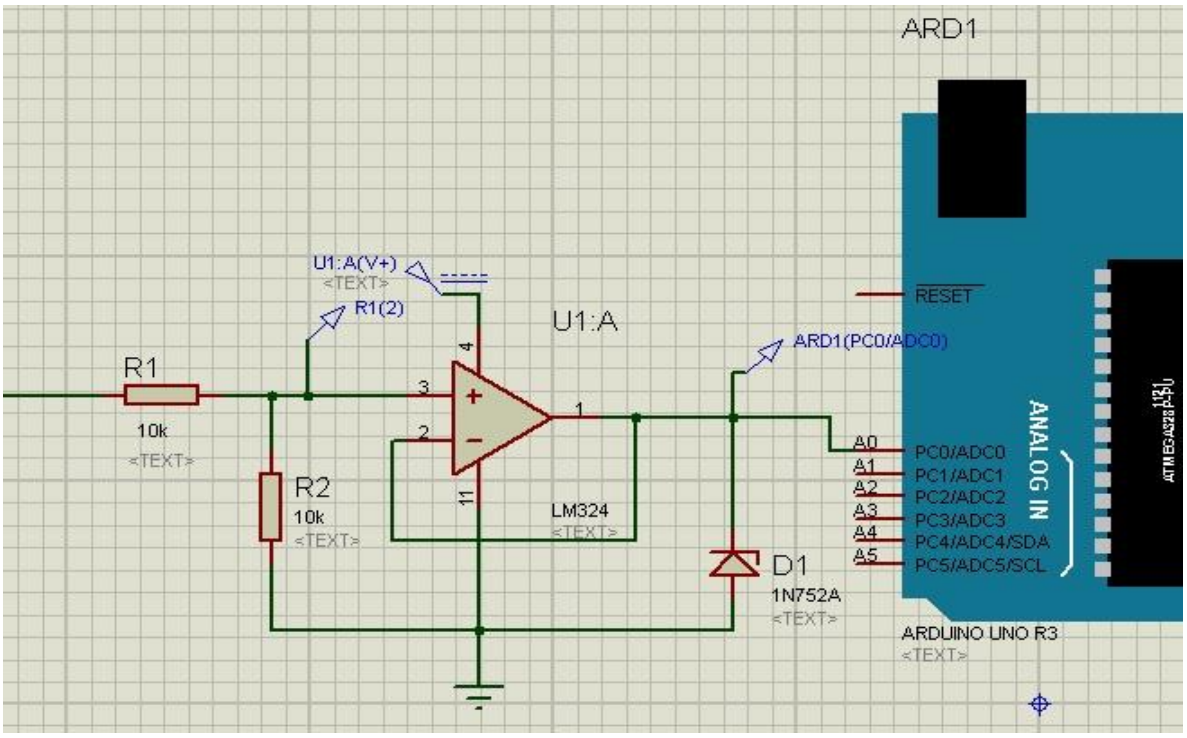


Figura 2. 2: Conexión de la entrada analógica al microcontrolador.

2.1.2 Diseño de las salidas analógicas de 0-10 V.

Al empezar a trabajar con el Arduino, se puede apreciar que la función analogWrite (función que permite actuar sobre las salidas analógicas) no proporciona realmente una tensión continua de salida, sino una señal de PWM. Esto se debe a que dicha función no está asociada a ningún conversor digital - analógico (D/A), o sea, el microcontrolador ATmega no tiene implementado conversores D/A en su interior, pero dependiendo de la resolución requerida la solución pasa por aprovechar la señal PWM [15].

Señal PWM

En una señal PWM la frecuencia base está fijada, pero el ancho del pulso es variable, o sea, que el ciclo de trabajo se puede hacer oscilar entre el 0% y el 100%, de acuerdo a la amplitud de la señal original. Para aprovechar esta señal en las aplicaciones que se presentan en este trabajo es necesario convertirla en una señal desmodulada, lo que se puede lograr con el empleo de un filtro, ver **Figura 2. 3**

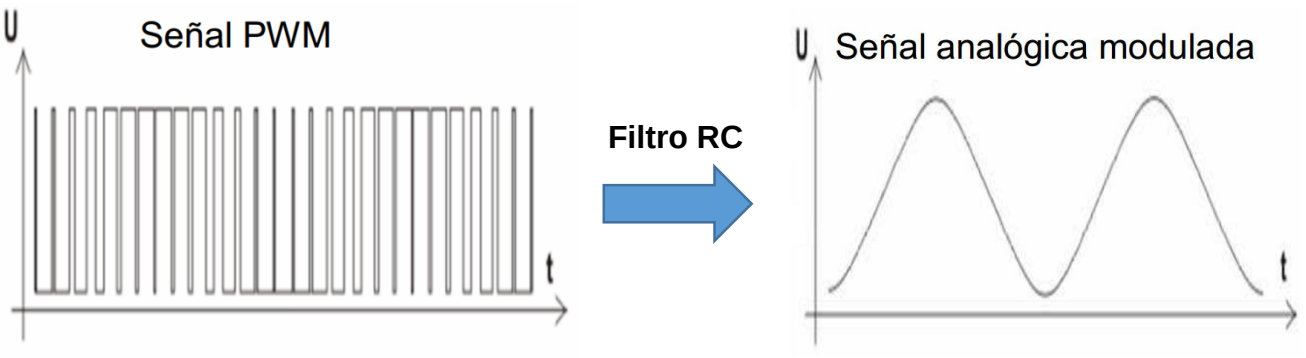


Figura 2. 3: Conversión de una señal PWM en analógica.

Un análisis de Fourier de una señal PWM muestra la existencia de un pico principal a la frecuencia Fn = 1/T, mostrando además la existencia de otros picos destacables a F = K/T, donde K es un entero. Los picos con K ≥ 2 son armónicos de la componente principal, los cuales son innecesarios y deben ser eliminados. Esto requiere que la señal PWM debe ser filtrada mediante filtros pasa bajos que puedan cancelar este ruido inherente a la propia señal [15].

En la **Figura 2. 4** se muestra el espectro de frecuencia de una señal PWM.

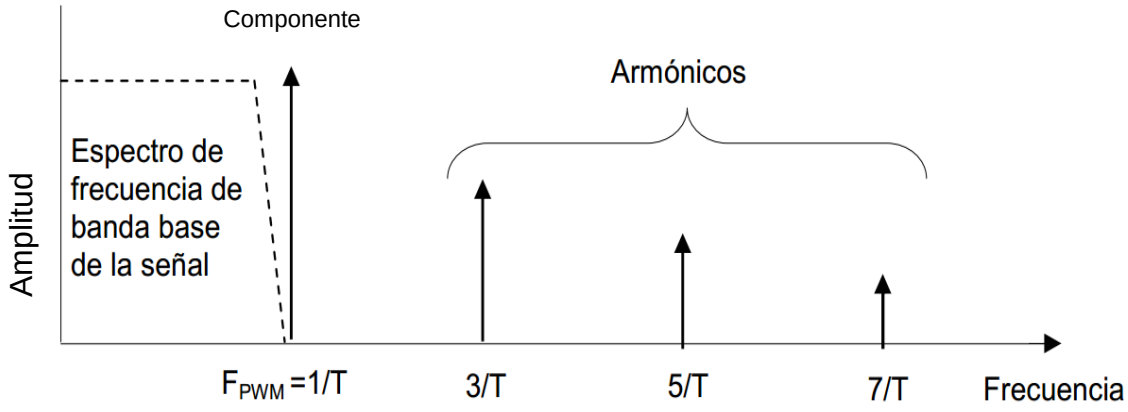


Figura 2. 4: Espectro de frecuencia de la señal PWM.

Frecuencia de los pines

Para el trabajo con el Arduino, se escribe un valor de 0 a 255 en un pin PWM, y la biblioteca de Arduino hace que este pin salida emita una señal PWM cuyo tiempo es proporcional al valor escrito.

Ahora bien cuando realmente llega el momento de establecer una tensión en algunos de los pines de salida, el valor 0-255 carece de significado debido a que lo que realmente se necesita en muchos casos es una tensión. Ahora suponiendo que el Arduino está funcionando a Vcc = 5 V entonces un valor de 255 también será igual a 5 V. Ahora bien se puede convertir fácilmente la tensión deseada al valor digital solo que es necesario el uso de una simple división. En primer lugar, se divide el voltaje deseado por el voltaje máximo (5 V). Esta división proporciona el porcentaje de la señal PWM. A continuación, se multiplica este porcentaje por 255 para obtener el valor deseado como se muestra en la fórmula para modular la señal (1).

Pin Valor (0-255) = 255 * (Voltaje Analógico / 5) (1)

La mayoría de los microprocesadores permiten cambiar la frecuencia de modulación para los pines PWM por tanto el Arduino también tiene sus propios valores por defecto. Valores que para los pines 3, 9, 10, 11 es de aproximadamente 488 Hz y para los pines 5 y 6, se trata de 977 Hz. Estos valores son para proporcionar una frecuencia determinada al Arduino mientras está funcionando a 16MHz. Estas frecuencias pueden cambiarse fácilmente escribiendo nuevos valores en el registro del temporizador apropiado. [21]

Por ejemplo, para cambiar la frecuencia del temporizador 2, que controla los pines 9 y 10, a una frecuencia de 3.906 Hz, se configuraría el registro, implementando este código (2) en el void setup del programa adios.pd cargado al IDE del Arduino.

TCCR1B= TCCR1B y 0b11111000 |0x02; (2)

Esta configuración del registro pudo validarse rigiéndose por los valores de la **Tabla 2. 1 y la Tabla 2. 2.**

Tabla 2. 1: Valores adjuntos de PWM para el temporizador 1 (pines 5 y 6.)

TCCR0B valor	Factor de preescala	Frecuencia
32(1)	1	62500
32(2)	8	7812.5
34	64	976.5625
35	256	244.140625
36	1,024	61.03515625

Tabla 2. 2: Valores adjuntos de PWM para el temporizador 2 (pines 9 y 10.)

TCCR1B valor	Factor de preescala	Frecuencia
1	1	312500
2	8	3906.25
3	64	488.28125
4	256	122.0703125
5	1,024	30.517578125

Filtro Pasa Bajo

Luego de analizar cómo funciona PWM e incluso como se puede cambiar la frecuencia de esta señal, es el momento de analizar la forma de convertir una señal de PWM en una tensión continua. Existe una manera fácil de desmodular esta señal de salida del Arduino. Para ello solo es necesario implementar un filtro pasa bajo de primer orden, para obtener una señal continua, y un amplificador para adaptarla al rango de voltaje requerido, como se muestra en la **Figura 2. 5**.

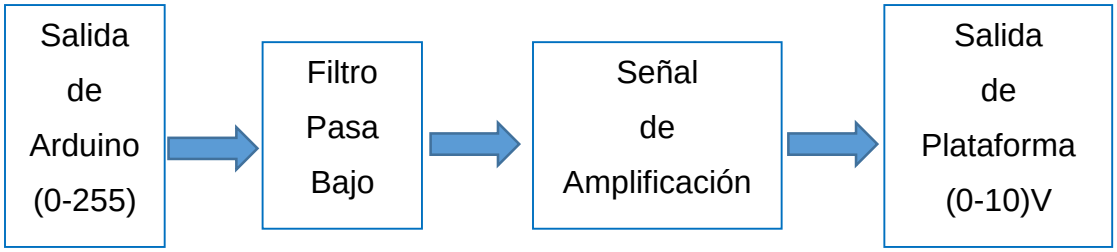


Figura 2. 5: Demodulación y amplificación de la señal de salida del Arduino.

En el esquema del filtro pasivo de primer orden pasa bajo que se muestra en la **Figura 2. 6**, cuando se aplica un voltaje a la entrada del circuito, el condensador (C) se comienza a cargarse. Ya cargado este, deja de conducir la corriente y la tensión en la salida del circuito coincide con la entrada (suponiendo una carga de alta impedancia). Ahora bien, recordando que los condensadores bloquean la corriente directa, pero dejan pasar corrientes alternas, se puede ver que cualquier entrada de voltaje de DC (directa) también será de salida, pero para altas frecuencias el voltaje de CA (alterna) se reducirá a tierra. Por tanto para señales de CA de menor frecuencia se produce el filtrado de acuerdo con la constante de tiempo R/C formado por la red resistencia-condensador [21].

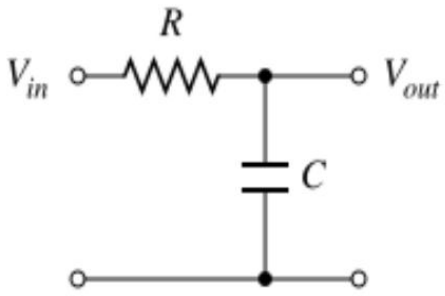


Figura 2. 6: Filtro pasa bajo RC.

Aunque este circuito es muy simple, para elegir los valores adecuados de R y C se deben tomar algunas decisiones de diseño, como son: la cantidad de ondulación que se puede tolerar y qué tan rápido lo hace el filtro responder. Estos dos parámetros son mutuamente excluyentes. En la mayoría de los filtros se espera alcanzar el filtrado ideal; que no es más que pasen todas las frecuencias inferiores a la frecuencia de corte, sin rizado de la tensión. Si bien no existe tal filtro ideal, se pude conseguir uno cercano de este mediante el uso de filtros de polos múltiples. Tales filtros podrían incorporar muchos componentes en una configuración de escalera. Mientras que el desarrollo de un filtro de orden superior proporciona muy buenas características de rendimiento, debido a su complejidad y costo no sería necesario para nuestra aplicación.

En la conversión de la señal PWM a una de DC se requiere lograr un buen compromiso entre baja frecuencia de corte y bajo rizado en la señal de salida. Para llegar a estas especificaciones, fue implementado un filtro de polo pasivo y simple**.** El inconveniente de realizar esta conversión con una frecuencia de corte baja es que en primer lugar, se limita la velocidad con la cual se puede variar la tensión de salida y en segundo lugar, que hay un retardo de la respuesta cuando se cambia el voltaje hasta que se alcanza la tensión de estado estacionario. Para muchas de las aplicaciones más comunes, esta disyuntiva es perfectamente aceptable [21].

Para la implementación del filtro pasa bajo se pudiera elegir una alta combinación entre condensador / resistencia, pero entonces significaría que se toma un largo tiempo para alcanzar el voltaje de salida adecuado, tiempo que sería similar al de carga del condensador. Y que además puede limitar en gran medida la rapidez con que la señal puede cambiar y ser vista en la salida. Por tanto, debe elegirse un valor de tensión de rizado razonable [21].

Para determinar el valor de los componentes del filtro, primero se elige el valor del condensador, el cual puede ser de 0.1uF. Luego se debe elegir el valor de la resistencia que se necesita, teniendo en cuenta que en las aplicaciones que en este trabajo se proponen, se requiere un tiempo de establecimiento del filtro (para alcanzar el 90% del valor final) de 0.1 segundos. Finalmente se escoge una resistencia de 48 kΩ, pues cumple con la exigencia antes mencionada, quedando el circuito de salida analógica como se muestra en la **Figura 2. 7**

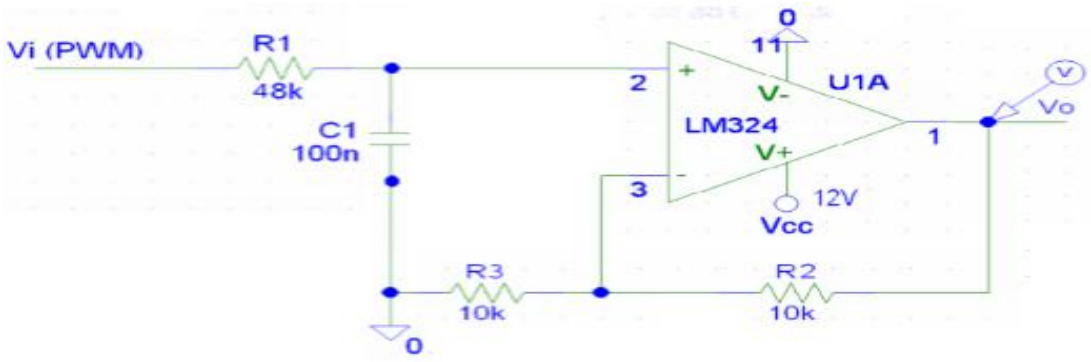


Figura 2. 7: Esquema amplificador de tensión.

En la **Figura 2. 8** se muestra el esquema completo de la salida analógica de la plataforma Arduino uno.

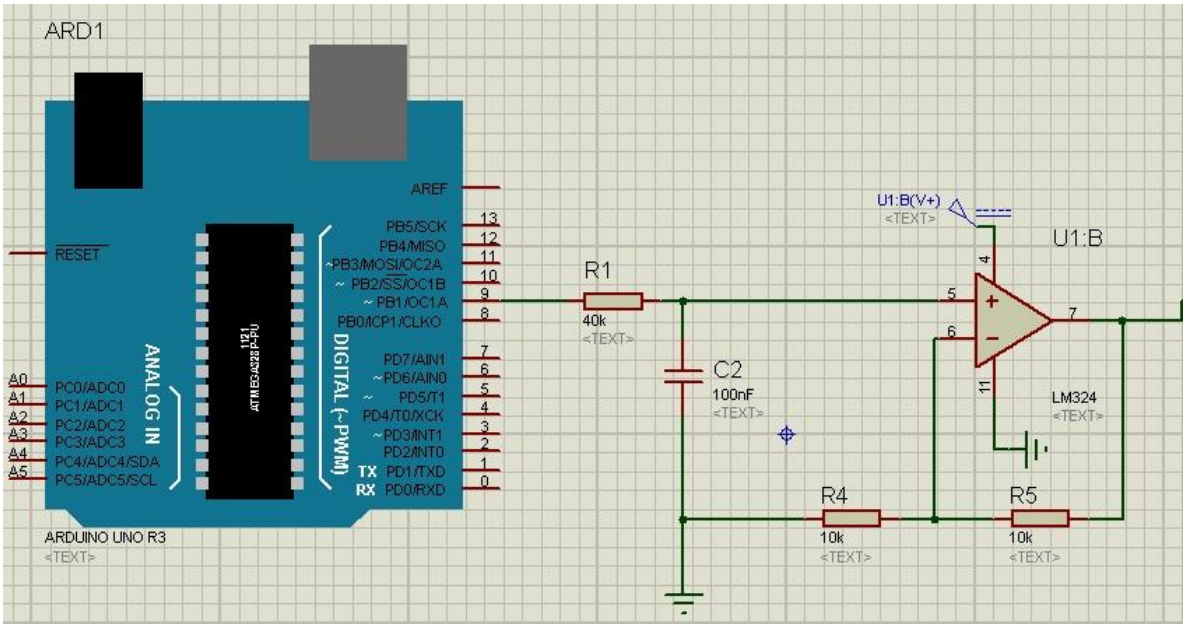


Figura 2. 8: Estructura de la salida analógica de la plataforma Arduino uno.

Luego de haber desarrollado el escudo de Arduino para poder garantizar el acondicionamiento previo de las señales de entrada y salida adaptadas a los rangos de 0 – 10V requeridos se muestra en la **Figura 2. 9**.

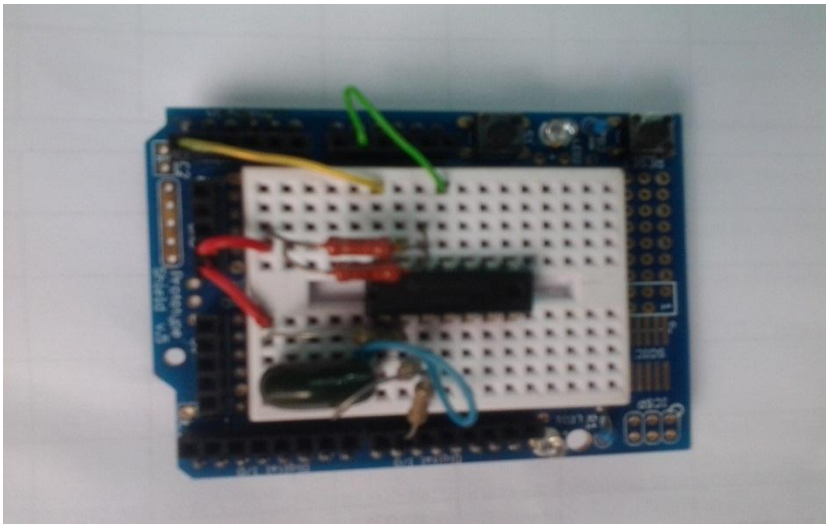


Figura 2. 9: Escudo de Arduino para adaptar las señales al rango requerido.

2.2 Interacción Arduino y MatLab.

MatLab es un programa de cálculo numérico que permite analizar datos, funciones, matrices y crear algoritmos y programas de manera sencilla y rápida a través de su lenguaje propio. Cuenta también con múltiples paquetes y herramientas adicionales que permiten ampliar sus posibilidades, cómo Mupad, una herramienta de cálculo simbólico o Simulink, que permite trabajar con sistemas físicos dinámicos.

Al controlar Arduino a través de MatLab se cuenta con más potencia para realizar cálculos matemáticos, lo cual es una muy buena idea, ya que sería de gran utilidad para la realización de operaciones matemáticas complejas o que requieran más capacidad de la que ofrece el microcontrolador de 8-bits. Además, se puede construir el programa “sobre la marcha”, lo cual es una ventaja muy importante debido a que es necesaria la realización de cambios en los programas realizados en tiempo real.

2.2.1 Pasos para la conexión Arduino – MatLab.

Preparativos para la conexión.

Para realizar la interacción Arduino- MatLab solo es necesario una placa Arduino (en nuestro caso la placa UNO r3) y un cable USB de tipo A a tipo B. En cuanto al software es obvio que se va a necesitar el MatLab, en el caso de nuestra aplicación, la versión MatLab R2012b y la IDE de Arduino. Además, también se requiere descargar los paquetes de soporte de Arduino para MatLab, luego deben ser descomprimidos y guardados en la carpeta ArduinoIO que va a crear en alguna dirección. Es posible que los paquetes de soporte no funcionen en versiones de MatLab como la R2010a o anteriores, por lo que es recomendable utilizar versiones de MatLab de la R2011a en adelante.

Arduino y el puerto USB

Es conectada la placa Arduino a la computadora y mediante la IDE de Arduino le es cargado el programa adioes.pde que se encuentra ubicado en la carpeta:
ArduinoIO/pde/adioes

Este programa es imprescindible para que Arduino sea capaz de reconocer los comandos que le envía MatLab.

MatLab

Asegurarse que el archivo **pathdef.m**, se encuentre ubicado dentro de la carpeta **toolbox/local de MatLab**.

Luego se copian los archivos arduino.m, contents.m e install_arduino.m desde ArduinoIO a la carpeta bin de Matlab (una vez más se modifican las rutas según sea necesario).

C:\Program Files\MATLAB\R2012b\toolbox\ArduinoIO

Finalmente solo queda ejecutar el MatLab y disfrutar de las bibliotecas de Simulink para Arduino

2.3 Prácticas de Laboratorio Diseñadas.

Práctica de laboratorio #1.

Título: Caracterización de la plataforma desarrollada como instrumento de medición

Objetivos:

1. Obtener las curvas características de entrada y salida de la plataforma desarrollada, comparada con un instrumento modelo.
2. Obtener el error fundamental tanto a la entrada como a la salida de la plataforma.

Bibliografía.

1. Salazar, A. Verificación de voltímetros y amperímetros con clase de precisión mayor que 1.
2. Amestegui, M. Apuntes de control PID. Universidad de San Andrés. Bolivia. Enero 2011.
3. Evans, M.; Noble, J.; Hochenbaum, J. Arduino in action. Editorial: Manning. ISBN: 9781617290244. 2013.
4. Enríquez, R. Guía de Usuario de Arduino. Universidad de Córdoba. 13 de noviembre de 2009.

Fundamentos teóricos.

Medición es la determinación del valor numérico de una magnitud física por medios experimentales empleando medios técnicos cifrados en valores de magnitudes físicas tomadas como unidades.

En metrología se asume que los errores de medición siempre están presentes en el resultado de las mediciones, independientemente del cuidado y la calificación del operador, así como de los medios principios y métodos de medición empleados.

Estos errores se deben al no perfeccionamiento de los medios y métodos de medición, así como a las condiciones ambientales, la posición de los instrumentos, el estado de los órganos de los sentidos del operador, etc.

El comportamiento de un sensor o de un instrumento de medida se puede definir, en general, mediante la curva de calibración (ver figura 2.10) que indica tanto el comportamiento en régimen estático como en dinámico. En régimen estático corresponde a la relación entre la entrada y la salida cuando la entrada es constante o cuando ha transcurrido un tiempo suficiente para que la salida haya alcanzado el valor final o régimen permanente. Por otra parte en régimen dinámico indica la evolución del sistema hasta que la salida alcanza el valor final ante la variación en la entrada.

Una curva de calibración en la que se obtenga con rigurosidad ambos comportamientos resulta tremendamente compleja por lo que, en la práctica suelen indicarse por separado mediante una serie de parámetros

Figura 2. 10: Ejemplos de curvas de calibración. **(a)** Curva de calibración lineal **(b)** Curva de calibración no lineal.

Errores de los instrumentos y formas de expresarlos

Los instrumentos de medición se fabrican para trabajar en condiciones determinadas, pero existen ocasiones que en las cuales no se cumplen esas condiciones, por eso de los errores de los instrumentos de acuerdo a las condiciones de trabajo se dividen en fundamentales y adicionales.

Error fundamental del instrumento: es el error del instrumento de medición cuando se emplea en condiciones normales.

Error adicional de instrumento: es la variación del error del instrumento de la medición producido, por la desviación de su valor normal, de una de las magnitudes que influyen sobre el instrumento, o porque la magnitud, cuyo valor se mide, se sale de los límites establecidos para el medio de medición.

El error del instrumento es la diferencia entre la indicación del instrumento X_I ; y el valor real X de la magnitud cuyo valor se mide. Este error de acuerdo a la forma de expresarlo se divide en absoluto Δ_X , relativo δ_X y relativo reducido ɣ

La definición de error absoluto se coincide con la del error del instrumento y se calcula

$$\Delta X = X_I - X \qquad (4)$$

El error relativo del instrumento de medición es la relación entre el error absoluto y el valor real de la magnitud que se mide, este error generalmente se expresa en por ciento y como la indicación del instrumento y el valor real son prácticamente iguales no se introduce error apreciable en el cálculo, si en lugar del valor real que generalmente no se conoce, se toma la indicación del instrumento.

$$\delta_X = \frac{\Delta X}{X_I} 100 \qquad (5)$$

El análisis anterior muestra que, que el error relativo depende del valor real de la magnitud sometida a medición, por eso en muchos casos no se puede determinar la clase de precisión del instrumento en base a este error, en estos casos la clase de precisión se determina en base al error relativo reducido ɣ, el cual está definido

como la relación entre el error absoluto y el valor normalizado X_N para cada medio de medición.

$$\gamma = \frac{\Delta X}{X_N} 100 \qquad (6)$$

Resultados de la verificación

- El error relativo reducido, el cual permite determinar la clase de precisión de los instrumentos verificados.
- El error relativo que es la componente de instrumental del error de las mediciones directas.
- El error absoluto que permite introducirle corrección al resultado de la medición.

Corrección C es la magnitud que se le suma algebraicamente a la indicación del instrumento.

Para obtener el valor real de la magnitud, la misma es numéricamente igual al error absoluto pero en signo contrario.

$$C=-\Delta X \qquad (7)$$

Es bueno aclarar, que al introducirle corrección a la indicación de un instrumento esta no queda libre de error, pues en la determinación del valor real con el instrumento modelo también se introduce error.

En las normas se establece, que cuando a un instrumento se le introduce conexión este se comporta como si fuera de la clase inmediata superior.

Para la implementación del programa en MatLab fueron utilizadas las Interfaz Graficas de Usuario (GUIDE) que es un entorno de programación visual disponible en MATLAB para realizar y ejecutar programas que necesiten ingreso continuo de datos.

La aplicación GUIDE consta de dos archivos: .m y .fig. El archivo .m es el ejecutable y el .fig la parte gráfica. Para la Interfaz Gráfica, simplemente se ejecuta en la ventana de comandos el nombre del .fig en el caso de esta práctica “interfaz”

>> interfaz.

En cuanto a la realización del programa fue necesario lograr cambiar el canal mediante el cual se estaba trabajando con el Arduino así como poder leer o escribir a través de los pines. Además de poder tabular todas las mediciones realizadas. Para el cumplimiento de estas tareas fue necesario la realización de los siguientes fragmentos de código.

- Para lograr la conexión del Arduino.

```
global a;                    %% declara una variable global para el
Arduino
%% Elección del canal a conectar el Arduino según la opción
escogida
switch canal
    case 1
     set(handles.text6,'string','Conectando a DEMO');
     a=arduino('DEMO');
     set(handles.text6,'string','Conectado');
    case 2
     set(handles.text6,'string','Conectando a COM0');
     a=arduino('COM0');
     set(handles.text6,'string','Conectado');
```

- Para lograr leer de un pin y llevarlo a valores de voltaje.

```
global a;                  %% declarar la variable global para el
Arduino
a.pinMode(5,'input');                %% poner el pin 5 en modo
lectura
av=a.analogRead(5);                  %% guarda en av la lectura del
pin 5
av=(av/1023)*5;                      %% se convierte la lectura de
0—1024
                                        %% a una de 0-5 V.
```

- Para lograr escribir en un pin.

```
a.pinMode(5,'output');                 %% poner el pin 5 en modo
escritura
sal = get(handles.slider2,'value');    %%toma el valor del slider y
guarda
a.analogWrite(5,round(sal));           %% escribe el valor en el
pin 5
```

- Para tabular los resultados

```
set(handles.uitable3,'Data',[]);  %% se envía a la tabla un
arreglo vacío
       %% se guarda en una variable el 1er valor de la tabla
valordeseado= [valordeseado;
str2double(get(handles.edit2,'string'))];
       %% se guarda en otra variable el 2do valor de la tabla
valorreal=[valorreal;str2double(get(handles.text1,'string'))];
       %% se envía a la tabla el arreglo con los 2 valores de la
tabla
set(handles.uitable3,'Data',[valorreal,valordeseado]);
```

La interfaz propuesta para la práctica docente se muestra en la **Figura 2. 11.**

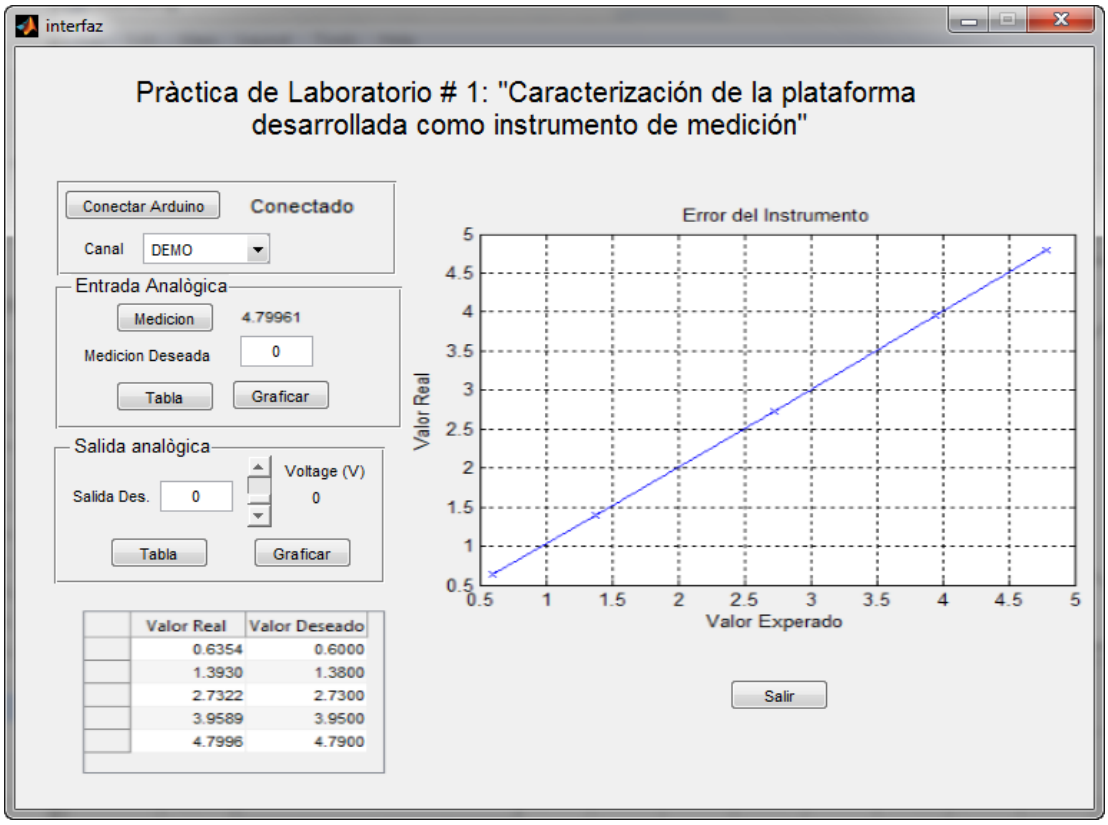


Figura 2. 11: Ventana de la interfaz gráfica propuesta para la práctica.

Instalación experimental

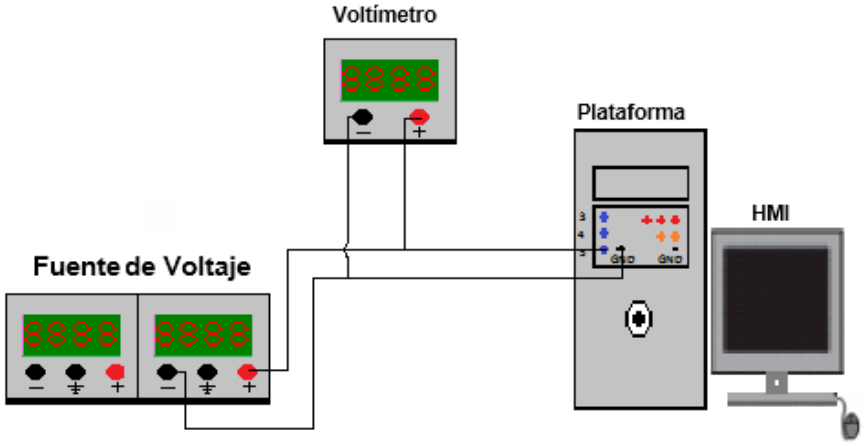


Figura 2. 12: Esquema de la instalación experimental de la práctica. (Verificación de salida de la plataforma).

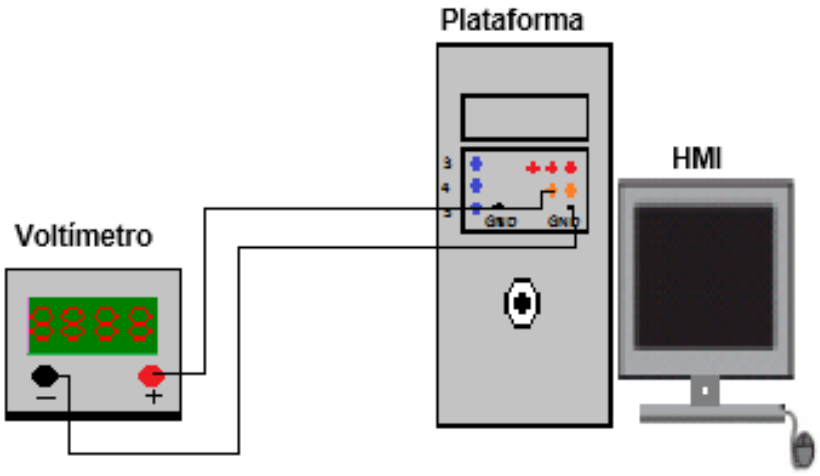


Figura 2. 13: Esquema de la instalación experimental de la práctica. (Verificación de entrada de la plataforma).

Indicaciones para el trabajo.

Determinación de las características estáticas de la entrada analógica de la plataforma desarrollada.

1. Encender la PC y correr la aplicación (GUIDE) relacionada con este laboratorio en el software MatLab. El profesor debe indicarle la dirección en la PC donde se guarda dicha aplicación.
2. Conectar una fuente de voltaje a una entrada de la plataforma desarrollada, empleando para ello la entrada A5 y tierra (verificar polaridad antes de encender).
3. Conectar un voltímetro modelo para medir el mismo voltaje.
4. Comenzando desde cero, realizar incrementos de 1V hasta llegar a 10V y medir en cada caso.
5. Cada valor medido por el voltímetro se debe introducir en la interface desarrolla en el GUIDE de MatLab. Esto se puede realizar en el espacio en blanco correspondiente a **Medición deseada,** el cual se encuentra dentro del cuadro **Entrada analógica**, luego se presiona el botón **Medición** para obtener el valor medido por la plataforma. Finalmente se presiona el botón **Tabla** para guardar ambos resultados en la tabla que se encuentra en la parte inferior del HMI.
6. Determinar los errores: absoluto, relativo y relativo reducido para cada medición.
7. Para obtener la gráfica de los valores reales vs valores deseados, al finalizar la introducción de todos los valores de la medición se presiona el botón **Graficar** que se encuentra dentro del mismo cuadro **Entrada analógica.**
8. El experimento se debe realizar también en forma descendente, por lo que se repiten los pasos 5 y 6 recogiendo los resultados de la medición desde el valor máximo (10V) hasta el mínimo (0V).
9. Determinar el error fundamental, que no es más que el mayor relativo reducido que se obtenga en cualquiera de los dos sentidos de calibración (ascendente y descendente).

Determinación de las características estáticas de salida analógica de la plataforma desarrollada.

1. Conectar el voltímetro directamente a la salida de la plataforma (S9 y tierra).
2. Desde el HMI se varían los valores en la salida de la plataforma desde 0V hasta 10V con incrementos de 1V, actuando sobre el slider que se encuentra

dentro del cuadro **Salida analógica**. Luego se toma el valor de la tensión a la salida medido por el voltímetro y se ingresa en el espacio en blanco correspondiente a **Salida Des.** Finalmente se presiona el botón **Tabla** para guardar ambos resultados en la tabla que se encuentra en la parte inferior del HMI.

3. Determinar los errores: absoluto, relativo y relativo reducido para cada medición.
4. Para obtener la gráfica de los valores reales vs valores deseados, al finalizar la introducción de todos los valores de la medición se presiona el botón **Graficar** que se encuentra dentro del mismo cuadro **Salida analógica.**
5. El experimento se debe realizar también en forma descendente, por lo que se repiten los pasos 2 y 3, recogiendo los resultados de la medición desde el valor máximo (10V) hasta el mínimo (0V).
6. Determinar el error fundamental.
7. Realizar informe de la práctica.

Contenido de informe.

1. Título de la práctica.
2. Objetivos de la práctica.
3. Esquema de la instalación experimental.
4. Tabla donde se contemplen los resultados de las mediciones realizadas durante la práctica.
5. Gráficos de las características estáticas obtenidas.
6. Cálculos realizados.
7. Conclusiones.

Práctica de laboratorio #2

Título: Control digital directo de motor de CD e identificación por el método gráfico

Objetivos:

1. Realizar el control digital directo de un motor de CD desde el MatLab Simulink usando la plataforma desarrollada.
2. Comprobar el efecto del filtrado de la señal medida mediante un filtro exponencial de primer orden.
3. Comprobar el efecto de las acciones proporcional, integral, derivativa y el efecto del tiempo de muestreo.
4. Aplicar entrada escalón para identificación por el método gráfico.

Bibliografía.

1. Señales y sistemas en tiempo discreto.
2. Gene F.; Franklin. Digital Control of Dynamic Systems
3. Amestegui, M. Apuntes de control PID. Universidad de San Andrés. Bolivia. Enero 2011.
4. Evans, M.; Noble, J.; Hochenbaum, J. Arduino in action. Editorial: Manning. ISBN: 9781617290244. 2013.
5. Enríquez, R. Guía de Usuario de Arduino. Universidad de Córdoba. 13 de noviembre de 2009.

Fundamentos teóricos.

El controlador PID (Proporcional, Integral y Derivativo) es un controlador realimentado cuyo propósito es hacer que el error en estado estacionario, entre la señal de referencia y la señal de salida de la planta, sea cero de manera asintótica en el tiempo, lo que se logra mediante el uso de la acción integral. Además el controlador tiene la capacidad de anticipar el futuro a través de la acción derivativa que tiene un efecto predictivo sobre la salida del proceso.

Los controladores PID son suficientes para resolver el problema de control de muchas aplicaciones en la industria, particularmente cuando la dinámica del proceso lo permite (en general procesos que pueden ser descritos por dinámicas de primer y segundo orden), y los requerimientos de desempeño son modestos (generalmente limitados a especificaciones del comportamiento del error en estado estacionario y una rápida respuesta a cambios en la señal de referencia) [1].

El control proporcional tiene la desventaja de que, en la mayoría de los casos, resulta en un error estático o de estado estacionario diferente de cero. Los algoritmos de control usados en la práctica son, por tanto, normalmente más complejos que el del controlador proporcional.

El algoritmo de control a utilizar corresponde a una versión en tiempo discreto del controlador PID continuo ideal (ISA) expresado por:

$$u(t) = K_p(e(t) + \frac{1}{T_I}\int_0^t e(\lambda)d\lambda + T_D\frac{de(t)}{dt}) \quad (8)$$

Que en versión discretizada se puede expresar como:

$$u_k = u_{k-1} + du_k;$$

$$du_k = q_0.e_k + q_1.e_{k-1} + q_2.e_{k-2}$$

O bien

$$u_k = u_{k-1} + q_0.e_k + q_1.e_{k-1} + q_2.e_{k-2}$$

Donde $u_k = u(k * T_0);$ $e_k = e(k * T_0);$ $k = 0,1,2, \ldots \ldots$

Siendo T_0 el período de muestreo.

En términos de función de transferencia se expresa:

$$G_c = \frac{U(z)}{E(z)} = \frac{1+q_0.z^{-1}+q_1.z^{-1}+q_2.z^{-1}}{1-z^{-1}} \quad (9)$$

Los parámetros del controlador discretos se relacionan con los del continuo por las siguientes fórmulas:

Si se aproxima por el método de los rectángulos:

$$q_0 = K_p(1 + \frac{T_D}{T_0})$$

$$q_1 = -K_p(1 - \frac{T_0}{T_I} + 2 * \frac{T_D}{T_0})$$

$$q_2 = K_p * \frac{T_D}{T_0}$$

Si se aproxima por el método de los trapecios:

$$q_0 = K_p(1 + \frac{T_0}{2 * T_I} + \frac{T_D}{T_0})$$

$$q_1 = -K_p(1 - \frac{T_0}{2 * T_I} + 2 * \frac{T_D}{T_0})$$

$$q_0 = K_p * \frac{T_D}{T_0})$$

Programación del controlador PID digital

Para el trabajo con el PID en el modelo Simulink propuesto (ver **Anexo**) para la realización de la práctica, fue necesario realizar la programación de un bloque S-Funtion de las librerías del Simulink en MatLab. Donde el código de programación se muestra a continuación:

```
float ContPID(int aut, float yr, float y, float *e, float *q, float umin, float umax,float *u){

 float du;

   e[0]=yr-y;

   du=q[0]*e[0]+q[1]*e[1]+q[2]*e[2];
```

```
   e[2]=e[1];
   e[1]=e[0];
   if (aut==1)  // modo automático
{*u=*u+du;
        if (*u<umin)  *u=umin;
        if (*u>umax)  *u=umax;
       }
      return (0);
      }
/*aut: si está en manual o automático.
yr: referencia
y: entrada
*e: arreglo de errores (e(k-2),e(k-1),e(k))
*q: arreglo de coeficientes q (q1,q2,q3)
  q[0]=kp*(1+td/t0);
  q[1]=-kp*(1-t0/ti+2*td/t0);
  q[2]=kp*(td/t0);
umin: salida mínima del controlador (0)
umax: salida máxima del controlador (10)
*u: dirección  donde se va aguardar la salida del controlador*/
//-----------Controlador por modelo PID -----
float ContPID(float kp, float ti,float td, float r,float w, float *e,float *u){
   float du;
```

```
float q[3];
q[0]=kp*(1+td/t0);
q[1]=-kp*(1-t0/ti+2*td/t0);
q[2]=kp*(td/t0);
e[0]=r-w;
du=q[0]*e[0]+q[1]*e[1]+q[2]*e[2];
//du = kp*e[0];
e[1]=e[0];
e[2]=e[1];
    *u=*u+du;
     if (*u<0)  *u=0;
     if (*u>10)  *u=10;
return (0);
}
```

Filtrado Digital.

Un filtro es un sistema o una red que cambia selectivamente la forma de onda, o las características amplitud-frecuencia o fase-frecuencia de una manera deseada. Además de que dependiendo de las variaciones de las señales de entrada en el tiempo y amplitud, se realiza un procesamiento matemático sobre dicha señal; generalmente mediante el uso de la Transformada rápida de Fourier; obteniéndose en la salida el resultado del procesamiento matemático o la señal de salida.

Los objetivos comunes del proceso de filtrado son mejorar la calidad de la señal, por ejemplo removiendo o atenuando el nivel de ruido, extrayendo información de dos o más señales previamente combinadas para hacer uso eficiente de un canal de comunicación, etc.

Los filtros digitales son usados generalmente para dos propósitos:

1. Separar señales que han sido combinadas, por ejemplo: el caso del latido del corazón de un niño en el vientre de la madre se combina con su respiración y latidos.
2. Restaurar señales que han sido distorsionadas en alguna manera. Los filtros analógicos se pueden usar para esas mismas tareas, no obstante los filtros digitales pueden lograr resultados muchos mejores.

Un filtro digital es un algoritmo implementado en hardware y/o software que opera sobre una señal de entrada digital (discreta en tiempo y cuantizada en amplitud) y genera una señal digital de salida, con la finalidad de efectuar un proceso de filtrado. El término “filtro digital” se refiere al hardware o software específico que ejecuta el algoritmo. Los filtros digitales trabajan sobre valores numéricos asociados a muestras de esas señales analógicas previamente digitalizadas por conversores A/D o simplemente sobre un conjunto de números almacenados en la memoria de una computadora o microprocesador.

Los filtros digitales juegan un papel muy importante en el procesamiento digital de señales. En gran número de aplicaciones, como compresión de datos, procesamiento de señales biomédicas, procesamiento de señales de voz, procesamiento de imágenes, transmisión de datos, audio digital, cancelamiento de ecos telefónicos, se prefieren por sobre los filtros analógicos por uno o más de los siguientes motivos:

- Los filtros digitales pueden tener características que son imposibles de conseguir con filtros analógicos, pueden incluso lograr un desempeño miles de veces superior a un filtro analógico.
- El desempeño de los filtros digitales no varía con las condiciones ambientales (temperatura, humedad, etc.) como sí ocurre con los filtros analógicos, lo que elimina la necesidad de calibrarlos periódicamente.
- Si el filtro se implementa utilizando un procesador programable la respuesta en frecuencia de un filtro digital puede ajustarse a voluntad (filtrado adaptivo).

El mismo filtro puede procesar varias señales o canales de entrada sin necesidad de replicar el hardware.

- Las señales filtradas y sin filtrar pueden almacenarse para uso o análisis posterior.
- Los filtros digitales pueden utilizarse a muy bajas frecuencias, como las que se encuentran en aplicaciones biomédicas, donde el empleo de filtros analógicos es poco práctico por los valores muy elevados de los componentes pasivos involucrados (capacitores, inductancias). Además, los filtros digitales pueden trabajar sobre un amplio rango de frecuencias simplemente cambiando la frecuencia de muestreo.

Sin embargo, los filtros digitales también presentan una serie de desventajas respecto a los filtros analógicos:

- Limitación de frecuencia. La frecuencia de Nyquist (fija el ancho de banda útil que el filtro puede procesar) queda definida por el proceso de conversión (tiempos de conversión del conversor A/D y D/A), velocidad del procesador, cantidad de operaciones a ejecutar por unidad de tiempo, etc.
- Efectos de longitud finita de palabra. En general, los coeficientes del filtro implementado serán distintos de los calculados teóricamente si la representación numérica que se utiliza para implementar el filtro no es de precisión infinita (punto flotante).No sólo influye la cuantización de los coeficientes del filtro, sino también el redondeo de las operaciones numéricas, la cuantización del conversor A/D y D/A, la truncación que ocurre al almacenar los contenidos del acumulador en memoria, etc.
- Tiempos de diseño y desarrollo prolongados. Los tiempos de diseño y desarrollo de un filtro digital, en especial el diseño del hardware pueden ser muy superior al necesario para el diseño de un filtro analógico El número de muestras anteriores a la actual que se utilizan en un filtro para generar una muestra de salida corresponde al orden del filtro. Un filtro de primer orden utiliza una sola muestra precedente.

Filtro exponencial de primer orden

R/(Ts+1)= Gf(s)

R = 1 para que no atenué ni amplifique.

Gf(z) = Z(Gf(s))

$$Gf(z) = \frac{(b_0 + b_1 z^{-1} + \cdots + b_n z^{-n})}{(1 - a_0 z^{-1} + a_1 z^{-2} + \cdots + a_m z^{-m})}; m > n$$

$$Y(k) = a_1 Y(k-1) + a_n Y(k-n) + b_0 X + b_1 X(k-1) + b_n X(k-n)$$

$\frac{Y(s)}{X(s)} = \frac{1}{\tau s+1}$ Antitrasformando τy' + y = x $Y = \frac{Y_{k1} - Y_{k1}}{T_0}$

Sustituyendo ⟶ $\tau\left(\frac{Y_{(k)} - Y_{(k-1)}}{T_0}\right) + Y_{(k)} = X_{(k)}$

$$Y(k) = \frac{1}{1 + \frac{T_0}{\tau}} * Y(k-1) + \frac{\frac{T_0}{\tau}}{1 + \frac{T_0}{\tau}} X(k); \qquad Y(k) = aY(k-1) + bX(k)$$

a b

$$Gf(z) = \frac{b}{1 - aZ^{-1}}; \quad b = 1 - a$$

$$Gf(z) = \frac{1 - a}{1 - aZ^{-1}}$$

Y(k) = a(Y(k-1)-X(k)) + X(k) 0 < a < 1

a = 1 --> Y(k) = Y(k-1) y el filtro filtra todo.

a = 0 --> Y(k) = X(k) y el filtro no filtra nada.

En cuanto a la realización del programa fue necesario lograr la interacción y el trabajo simultáneo de las GUIDE con el Simulink. Interacción que se logró conseguir mediante los siguientes pasos. Primeramente guardar el modelo Simulink en la

misma carpeta contenedora del programa. Luego en el .m del programa realizado escribir los siguientes fragmentos de código.

- Para abrir el Modelo Simulink

```
find_system('Name','filtra2');  % Encuentra el Modelo Simulink
'filtra2'  open_system('filtra2');          % Abre el  modelo
Simulink
set_param(ModelName, 'SimulationCommand', 'start');  % Arranca el
modelo
```

- Para establecer (actualizar) parámetros en la interfaz

```
SP=str2double(get(handles.text1,'string')) % Se toma el valor y se
conv
   assignin('base','SP',SP)                % Se guarda en el
Workspace

     find_system('Name','filtra2');
     open_system('filtra2');
     set_param('filtra2/SP','Gain','SP'); % Se envia el valor con el
nombre del bloque que se quiere modificar y el nombre del modelo
Simulink
```

Instalación experimental

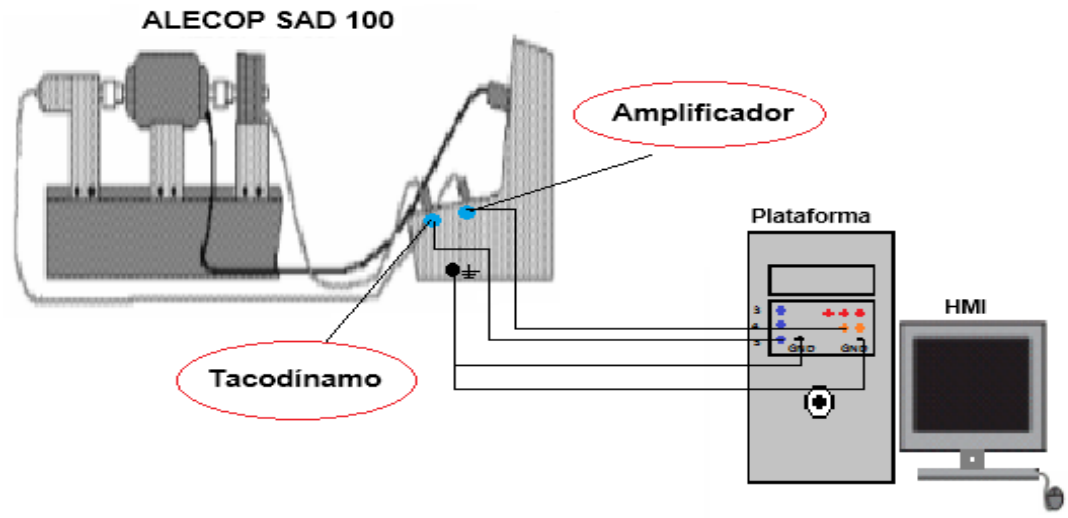


Figura 2. 14: Esquema de la instalación experimental de la práctica.

En la **Figura 2. 15** se muestra la interfaz gráfica para la práctica propuesta así como el código del programa desarrollado en el **Anexo 3.**

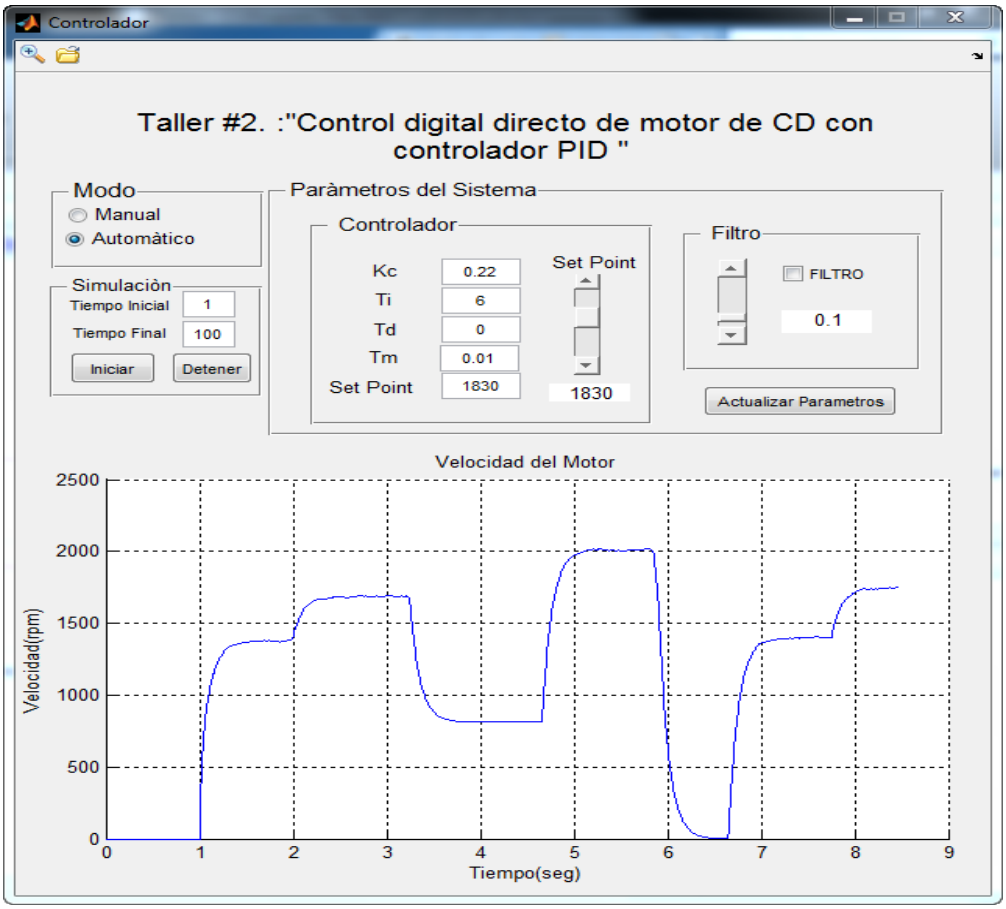


Figura 2. 15: Ventana de la interfaz gráfica propuesta.

Indicaciones para el trabajo.

1. Conectar la maqueta SAD-100 a la plataforma desarrollada. (Verificar conexiones antes de encender).
2. Desde el MatLab seleccionar el modo en que se desea trabajar (Automático/Manual).

Trabajando en modo Manual.

1. Ingresar valores en los espacios en blanco correspondiente a **Tiempo Inicial** y **Tiempo Final** de cuadro **Simulación,** para definir los tiempos de arranque y parada de la simulación.
2. Ingresar un valor en el espacio en blanco correspondiente **Set Point** o actuar sobre el slider que se encuentra dentro del cuadro **Controlador,** para establecer diferentes valores de velocidad en el motor.

3. Describir el comportamiento de la velocidad del motor.
4. Ingresar un valor en el espacio en blanco o actuando sobre el slider **Constante del filtro** que se encuentra dentro del cuadro **Filtro,** comprobar el efecto del filtrado y los cambios en la respuesta temporal para valores del coeficiente del filtro próximos a cero y próximos a 1.
5. Después de haber fijado todos los parámetros del sistema se presiona el botón **Actualizar parámetros** para actualizar cualquier cambio aplicado al sistema.
6. Elaborar hipótesis de posible causa del ruido.

Trabajando en modo Automático.

1. Ingresar valores en los espacios en blanco correspondiente a **Tiempo Inicial** y **Tiempo Final,** para definir los tiempos de arranque y parada de la simulación.
2. Ingresar un valor en el espacio en blanco correspondiente **Set Point** o actuar sobre el slider que se encuentra dentro del cuadro **Parámetros de Controlador,** para establecer diferentes valores de velocidad en el motor.
3. Utilizar valores de ajuste del PID por defecto.
4. Ingresar valores en los espacios en blanco correspondiente a **Kc, Ti, Td** y **Tm**, para comprobar el efecto de las acciones PID y el tiempo de muestreo en la dinámica del proceso.
5. Proponer mejoras al ajuste del controlador.
6. Mediante un freno, aplicar perturbación a la planta y comprobar el efecto del controlador.
7. Enviar señal tipo escalón a la planta y graficar la salida y la entrada de la misma para una identificación por el método gráfico.
8. Después de haber fijado todos los parámetros del sistema se presiona el botón **Actualizar parámetros** para actualizar cualquier cambio aplicado al sistema.

Práctica de laboratorio #3

Título: Interacción PC-Plataforma en la emulación de procesos simples.

Objetivos:

1- Desarrollo de habilidades en el control de procesos físicos con el empleo de la interacción PC-Plataforma.

Bibliografía.

1. Torregrosa, J.; Conceptos básicos de simulación de procesos. Universidad Politécnica de Valencia.
2. Proenza, R. Detección de fallos desde SCADA inteligente. Proyecto Final de carrera, Ingeniería de Control Automático. Universidad Oriente.
3. Amestegui, M. Apuntes de control PID. Universidad de San Andrés. Bolivia. Enero 2011.
4. Evans, M.; Noble, J.; Hochenbaum, J. Arduino in action. Editorial: Manning. ISBN: 9781617290244. 2013.
5. Enríquez, R. Guía de Usuario de Arduino. Universidad de Córdoba. 13 de noviembre de 2009.

Fundamentos teóricos.

La simulación de procesos se realiza mediante software donde se implementan los modelos que describen los procesos químicos, físicos, biológicos, así como otros procesos técnicos y operaciones unitarias. Los requisitos básicos para su aplicación requieren un conocimiento profundo de las propiedades químicas y físicas de los componentes puros y mezclas, de las reacciones, y de los modelos matemáticos que, en combinación, permiten el cálculo de un proceso usando la computación.

Es sabido que en la simulación convergen diversas corrientes del saber, como es el análisis de los métodos numéricos para la solución de ecuaciones tanto algebraicas como diferenciales, el modelado de procesos, operaciones unitarias y fenómenos de transporte, estimación de propiedades fisicoquímicas, etcétera.

Utilidad de la simulación de procesos

La simulación de procesos es una herramienta que se ha hecho indispensable para la solución adecuada de los problemas de procesos simples e industriales. La simulación, es una herramienta que permite analizar, diseñar y optimizar procesos de interés en muchas tareas multidisciplinares.

Los tres tipos de modelos que pueden resolverse por medio de la simulación son:

- Análisis de un proceso: sirve para predecir el comportamiento óptimo del proceso mediante la computación de las ecuaciones de diseño obtenidas a partir del modelado previo del mismo. Además permite la interpolación y extrapolación dentro de ciertos límites, así como la búsqueda de las condiciones fuera de la gama de propiedades conocidas.
- Diseño del proceso: la simulación proporciona todos los datos de proceso requeridos para el diseño detallado de los diferentes equipos y para la construcción de plantas a nivel banco, piloto o industrial, que después de construirlas y operarlas servirán para retroalimentar el modelo utilizado o para validarlo.
- Optimización del proceso: facilita la optimización del modelo de acuerdo a los datos experimentales obtenidos de la observación del proceso. En la simulación de procesos siempre se utilizan modelos que introducen aproximaciones y suposiciones, pero facilitan la descripción de una variable en un amplio intervalo de propiedades que puede no estar cubierto por los datos reales.

Software de simulación

El software de simulación de procesos describe, de manera más o menos explícita, los procesos en forma de diagramas de flujo donde las operaciones unitarias se colocan y se conectan las corrientes de intercambio de materiales y productos. El software tiene que ser capaz de resolver los balances de masa y energía para encontrar un punto de funcionamiento estable. El desarrollo de modelos para una mejor representación de los procesos reales es el núcleo del desarrollo del software

de simulación. El desarrollo del modelo implica la participación no sólo de la ingeniería química o la ambiental, sino también en la ingeniería de control así como de las técnicas de simulación matemática. La simulación de procesos es, por tanto, uno de los pocos campos donde los científicos y profesionales de la química, la física, la informática, las matemáticas, y de varios campos de la ingeniería trabajan juntos.

Los simuladores se pueden clasificar en:

- Simuladores con modelos previamente programados: en esto simuladores, el usuario utiliza paquetes de software de uso específico (los elaborados para una operación unitaria específica y un determinado rango de operación o de uso general que contienen en su estructura varias operaciones unitarias las cuales pueden ser interrelacionados entre sí para simular un proceso por ejemplo Aspen Hysys®, Aspen Plus®, Chemcad®, COCO®, etc.) que ya incluyen la programación del modelo. Aunque son poco versátiles (ya que funcionan como cajas negras) se trata de programas muy confiables y robustos, donde el usuario no tiene porqué ser un especialista en simulación. En general, el valor de las licencias es elevado, aunque existen programas gratuitos.
- Simuladores programables: en estos, el usuario programa sus propios modelos mediante el uso paquetes de software de cálculo matemático (Mathematica®, Matlab®, Scilab®, etc.). Esto permite al usuario aplicar sus propios modelos e interaccionar de una forma más profunda con los mismos. Se dispone de una mayor libertad para simular aunque como contrapartida requiere tanto de un mayor conocimiento de los principios científicos como de manejo de lenguajes de programación. El general, el valor de las licencias es accesible y existen programas muy fiables y gratuitos.

Descripción del proceso.

El sistema de nivel se compone de dos tanques en cascada, lo que representa un modelo de segundo orden con una entrada. El flujo de salida del tanque 1 cae en forma de cascada al tanque 2. Este a su vez tiene una salida mediante la cual el

agua cae directamente al depósito. Una bomba es responsable de tomar el agua desde el depósito hasta el tanque 1, convirtiéndolo en un proceso cíclico. La **Figura 2. 16** muestra el sistema de dos tanques.

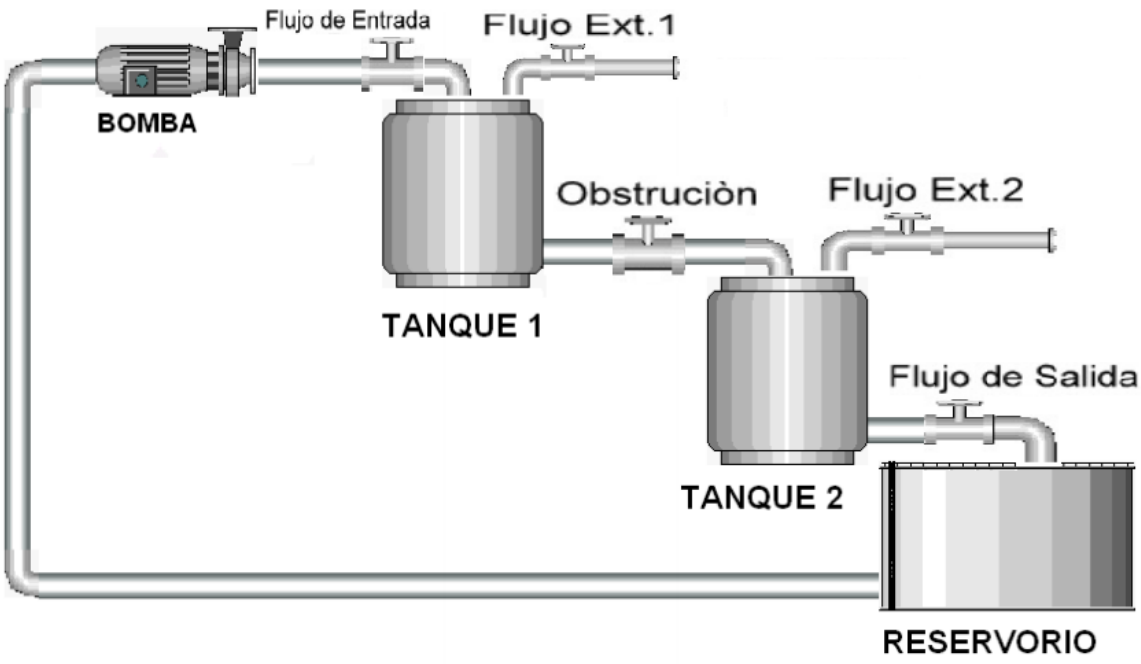


Figura 2. 16: Variables manipuladas y perturbaciones en Sistema de dos tanques.

El sistema de los dos tanques está basado en el control de nivel en el tanque 2. Este nivel depende del flujo de entrada al tanque 2, que a su vez guarda estrecha relación con el nivel en el tanque 1 (a mayor nivel en tanque 1 mayor flujo de entrada al tanque 2). La señal de control de la variable actúa sobre el voltaje de alimentación de la bomba que es directamente proporcional al flujo de entrada del tanque 1, actuando de forma directa en el nivel del primer tanque.

Para simular la dinámica de la planta, se utilizó un modelo matemático no lineal como se describe en las ecuaciones 10 y 11. La ecuación 10 calcula el nivel del tanque 1 y la ecuación 11 calcula el nivel del tanque 2.

$$L_1 = -\frac{a_1}{A_2}\sqrt{2gL_1} + \frac{K_m}{A_1}V_p \qquad (10)$$

$$L_2 = -\frac{a_2}{A_2}\sqrt{2gL_2} + \frac{a_1}{A_1}\sqrt{2gL_1} \qquad (11)$$

La **Tabla 2. 3** presenta una descripción de los parámetros mostrados en las ecuaciones 10 y 11, así como sus valores.

Tabla 2. 3: Parámetros de la planta.

Nombre	**Descripción**	**Valor**
K_m	Constante de la bomba	$4.6(cm^3/s)/V$
V_p	Tensión aplicada a la bomba	$0V < V_p < 22V$
a_1	Diámetro de salida del Tanque 1	0.178139cm
a_2	Diámetro de salida del Tanque 2	0.178139cm
A_1	Área del Tanque 1	$15.5179cm^2$
A_2	Área del Tanque 1	$15.5179cm^2$
g	Constante aceleración de la gravedad	$980\ cm/s^2$

Simulación de perturbaciones en tanque 1 y 2.

Tal como se muestra en la **Figura 2. 16**. La variable manipulada 1 (flujo de entrada), la perturbación 1 (flujo extra en el tanque 1) y la perturbación 2 (flujo extra en el tanque 2), influyen positivamente en el cambio del nivel del tanque 1 y del tanque 2 respectivamente, por otra parte la perturbación 3 (flujo de salida) actúa de forma negativa en el cambio de nivel del tanque 2. No obstante las unidades de medida que representan estas variable y perturbaciones son las mismas que las del proceso (ver **Tabla 2. 3**), no siendo así para la variable manipulada 2 (obstrucción a la salida del tanque 1) que se simuló en por ciento, disminuyendo así el área de salida del tanque 1. En los **Anexos** se muestran los modelos de Simulink implementados para simular las perturbaciones del proceso y el código del programa desarrollado respectivamente, así como en la **Figura 2. 17** se muestra la interfaz propuesta para la práctica.

En la **Figura 2. 17**se muestra la interfaz gráfica para la práctica propuesta, así como el modelo Simulink del proceso en el **Anexo.**

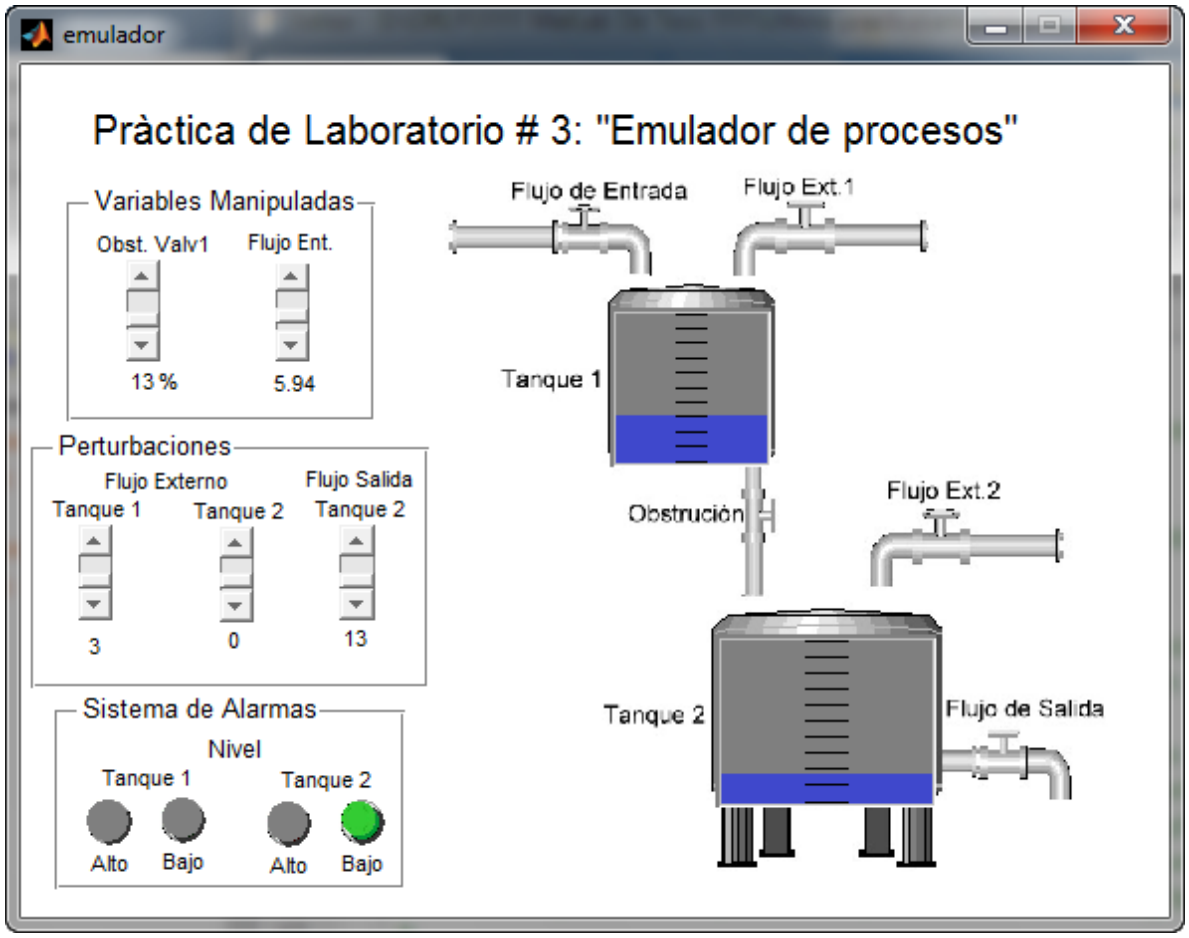


Figura 2. 17: Ventana de la interfaz gráfica propuesta.

Indicaciones para el trabajo.

1. Conectar instrumento de medición (fuente de voltaje) a la plataforma, según se muestra en el esquema de la instalación experimental desarrollada. Verificar conexiones antes de encender.
2. Variar el voltaje dentro del rango de 0-10V en las entradas analógicas A5 y A4 de la plataforma para cambiar el nivel en los tanques 1 y 2 respectivamente.
3. Observar el efecto causado por alguna perturbación aplicada actuando sobre los slider: **Flujo Ext. Tanque 1, 2** y **Flujo de Salida Tanque 2** que se encuentra dentro del cuadro **Perturbaciones**.
4. Proponer sistemas de control para la planta emulada.

Valoración económica y medioambiental.

Valoración económica.

Las prácticas de laboratorios implementadas se realizaron utilizando la plataforma Arduino Uno y con componentes eléctricos que se conectan a la misma. Los precios en el mercado internacional de la plataforma hacen que estas prácticas sean costeables y puedan desarrollarse en países en vías de desarrollo. En la **Tabla 2. 4** se recogen los precios de los componentes utilizados para la implementación de dicha plataforma.

Tabla 2. 4: Componentes de la plataforma desarrollada y costo de los mismos.

Descripción	**Unidades**	**Costo USD**
Arduino Uno	1	5.00
PCB Shield para Arduino	1	3.46
Resistencia	5	0.24
CI LM324	1	0.30
Tapa de Computadora	1	0.00
Cables y cabezales	10	0.00
TOTAL		**$9.00**

Conclusiones Parciales

En este capítulo se arribaron a las siguientes conclusiones:

1. Se confeccionó una plataforma de Arduino que consta de 3 entradas y 2 salidas analógicas, así como 3 salidas digitales y 2 tierras.
2. Se diseñaron tres prácticas de laboratorio docente utilizando la plataforma desarrollada, aprovechando su interacción con MatLab.
3. Se implementaron mediante las GUIDE de MatLab las interfaces gráficas correspondientes a las tres prácticas de laboratorio diseñadas.

Conclusiones Generales

Como conclusiones finales del presente trabajo de diploma se pueden citar las siguientes:

- La evaluación de las características de la plataforma Arduino demostró la factibilidad de su empleo en el desarrollo de la plataforma para el diseño o modernización de las prácticas de laboratorio de la carrera Ingeniería Automática.
- La interacción Arduino-MatLab permite aprovechar las ventajas del hardware y el software antes mencionados con fines docentes a través de prácticas de laboratorio, en la enseñanza de la especialidad de la Ingeniería Automática.
- Se confeccionó una plataforma de Arduino que consta de 3 entradas y 2 salidas analógicas, así como 3 salidas digitales y 2 tierras, para utilizarla en diferentes prácticas de laboratorio interactuando con el software MatLab, lo que permite introducir tecnologías modernas en el proceso enseñanza-aprendizaje de los estudiantes, aumentando así sus motivaciones.
- Se diseñaron tres prácticas de laboratorio con el empleo de la plataforma desarrollada, una de las cuales tributa a la disciplina de Instrumentación y el resto a la disciplina de Sistemas de control.

Recomendaciones

- Diseñar nuevas prácticas de laboratorio donde se aprovechen las ventajas de la plataforma desarrollada en la enseñanza de la especialidad de Ingeniería Automática.
- Utilizar la plataforma desarrollada para emular procesos más complejos.

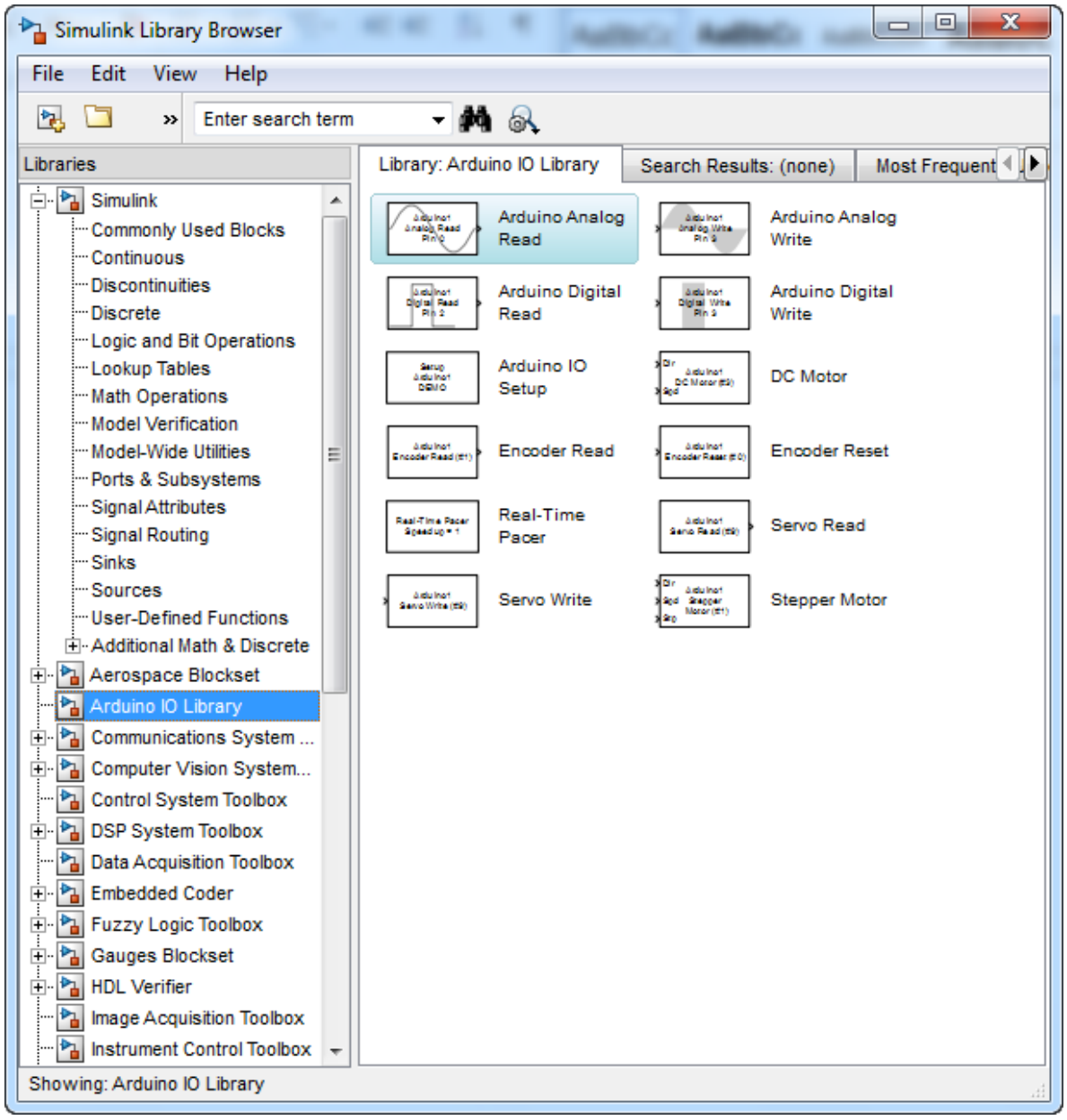
Bibliografía

1. Aladro, J. Cursillo de Electrónica Práctica. Club de Electrónica del Centro de Residencias de Eibar. 2000.
2. Amestegui, M. Apuntes de control PID. Universidad de San Andrés. Bolivia. Enero 2011.
3. Aprenda Robótica con Arduino. Curso Introductorio. Realizado por Colegio Santa Emilia. Basado en: www.arduino.cc, www.viaspositronicas.blogspot.com y www.earthshineEelectronics.com. 2013.
4. Bassi, E. Arduino Projects Book. Projects and text by Scott Fitzgerald and Michael Shiloh Additional text review by Tom Igoe. Italy. May 2013.
5. Campo, J.; y otros. Instrumentación Electrónica. Editorial: Thomson. ISBN: 84- 9732-166-9. España. 2003.
6. Clavijo, J. Diseño y simulación de sistemas microcontroladores en lenguaje C.
7. Enríquez, R. Guía de Usuario de Arduino. Universidad de Córdoba. 13 de noviembre de 2009.
8. Evans, M.; Noble, J.; Hochenbaum, J. Arduino in action. Editorial: Manning. ISBN: 9781617290244. 2013.
9. Fernández, J. Ejemplo de aplicación con Arduino: Medida de Caudal. Proyecto Final de Carrera, Ingeniería Técnica Industrial en Electronica Industrial. Escola Tècnica Superior d' Enginyeria. Universitat Politècnica de València. Diciembre del 2012.
10. Gil, J.E. Muñoz, A.J. Torres, V. Gómez, J.M. Uso de Simulink y Arduino para Prácticas de Robótica.
11. Legrá, Caleb; Utilización de la plataforma de Arduino Uno con fines docentes. Tesis en opción al título de Ingeniero en Automática. Universidad de Oriente. Cuba. (2015)
12. Lledó, E. Diseño de un sistema de control domótico basado en la plataforma Arduino. Proyecto Final de Carrera, Ingeniería Técnica en Informática de Sistemas. Escola Tècnica Superior d' Enginyeria Informàtica Universitat Politècnica de València. Diciembre del 2012.
13. Margolis, M. Arduino cookbook.
14. Minakawa, R.; y otros. Curso introductorio de Arduino. Grupo de Robótica en la Universidad Federal de Mato Grosso do Sul (UFMS). 2012.
15. Miras, J. Diseño de un extensor de entradas y salidas analógicas por MODBUS RTU sobre RS – 485. Ingeniería técnica de telecomunicaciones. Universidad Oberta de Catalunya. Febrero 2013.
16. Pereira, E.; Motta, M. Apostila Arduino. Curso de Ingeniería en Telecomunicaciones. Universidad Federal Fluminense. Diciembre 2010.
17. Plan de estudios D. Carrera de Ingenieria Automatica. Julio 2007.
18. Proenza, R. Detección de fallos desde SCADA inteligente. Proyecto Final de carrera, Ingeniería de Control Automático. Universidad Oriente.

19. Ruiz, J. Prácticas con Arduino Nivel I. Web Oficial de Arduino.
20. Salazar, A. Verificación de voltímetros y amperímetros con clase de precisión mayor que 1.
21. Scott, D. Arduino's AnalogWrite – Converting PWM to a Voltage. Junio 15 de 2011.

Anexos

Anexo 1: Biblioteca de bloques de Simulink para Arduino



Anexo 2: Código del programa de la Práctica 1

```
% --- Se ejecuta en cuanto carga el programa.
function interfaz_OpeningFcn(hObject, eventdata, handles, varargin)

global valordeseado;
global valorreal;
global valordeseado1;
global valorreal1;
global sal;
sal=0;
valordeseado=[];
valorreal=[];
valordeseado1=[];
valorreal1=[];
handles.vector = [];

set (handles.slider2,'value',0);
set(handles.text5,'String',mat2str(0));
set(handles.pushbutton1,'enable','off');
set(handles.pushbutton3,'enable','off');
set(handles.pushbutton4,'enable','off');
set(handles.pushbutton8,'enable','off');
set(handles.pushbutton7,'enable','off');
set(handles.pushbutton10,'enable','off');

% --- Se ejecuta en cuanto se presiona el Botón Conectar Arduino.
function pushbutton2_Callback(hObject, eventdata, handles)
global a;
global canal;

delete(instrfind('Type', 'serial'));  %% Desconecta el puerto del Arduino

%% Habilita los botones
set(handles.pushbutton1,'enable','on');
set(handles.pushbutton3,'enable','on');
set(handles.pushbutton4,'enable','on');
set(handles.pushbutton8,'enable','on');
set(handles.pushbutton7,'enable','on');
set(handles.pushbutton10,'enable','on');

%% Elección del Puerto a conectar el Arduino
switch canal
    case 1
     set(handles.text6,'string','Conectando a DEMO');
     a=arduino('DEMO');
     set(handles.text6,'string','Conectado');
    case 2
     set(handles.text6,'string','Conectando a COM0');
     a=arduino('COM0');
     set(handles.text6,'string','Conectado');
    case 3
     set(handles.text6,'string','Conectando a COM1');
     a=arduino('COM1');
     set(handles.text6,'string','Conectado');
    case 4
```

```
     set(handles.text6,'string','Conectando a COM2');
     a=arduino('COM2');
     set(handles.text6,'string','Conectado');
    case 5
     set(handles.text6,'string','Conectando a COM3');
     a=arduino('COM3');
     set(handles.text6,'string','Conectado');
    case 6
     set(handles.text6,'string','Conectando a COM4');
     a=arduino('COM4');
     set(handles.text6,'string','Conectado');
    case 7
     set(handles.text6,'string','Conectando a COM5');
     a=arduino('COM5');
     set(handles.text6,'string','Conectado');
end


% --- Se ejecuta en cuanto se presiona el Botón Medir.
function pushbutton1_Callback(hObject, eventdata, handles)
global a;
a.pinMode(5,'input');               %% pone el pin 5 en modo lectura
av=a.analogRead(5);                 %% guarda en av la lectura del pin 5
av=(av/1023)*5;                     %% se convierte la lectura de 0--1024
set(handles.text1,'String',av);     %% se muestra la lectura en el text1

% --- Se ejecuta en cuanto se presiona el Botón Tabla del Voltímetro.
function pushbutton3_Callback(hObject, eventdata, handles)
global valordeseado;
global valorreal;
set(handles.uitable3,'Data',[]);
valordeseado= [valordeseado; str2double(get(handles.edit2,'string'))];
valorreal=[valorreal;str2double(get(handles.text1,'string'))];
set(handles.uitable3,'Data',[valorreal,valordeseado]);

% --- Se ejecuta en cuanto se presiona el Botón Graficar del Voltímetro.
function pushbutton4_Callback(hObject, eventdata, handles)
global valordeseado;
global valorreal;
axes(handles.axes2)
cla(handles.axes2)
plot(valordeseado,valorreal,'-x');
grid on;
title('Error del Instrumento')
xlabel('Valor Experado')
ylabel('Valor Real')


% --- Se ejecuta en cuanto se desplaza la barra de la fuente.
function slider2_Callback(hObject, eventdata, handles)
global sal;
global a;
a.pinMode(5,'output');
sal = get(handles.slider2,'value');
a.analogWrite(5,round(sal));
set(handles.text5,'String',sal);
```

```
% --- Se ejecuta en cuanto se presiona el Botón Tabla de la fuente.
function pushbutton8_Callback(hObject, eventdata, handles)
global valordeseado1;
global valorreal1;
set(handles.uitable3,'Data',[]);
valordeseado1=[valordeseado1;str2double(get(handles.edit4,'string'))];
valorreal1=[valorreal1;str2double(get(handles.text5,'string'))];
set(handles.uitable3,'Data',[valorreal1,valordeseado1]);

% --- Se ejecuta en cuanto se presiona el Botón Graficar de la fuente.
function pushbutton10_Callback(hObject, eventdata, handles)
global valordeseado1;
global valorreal1;
axes(handles.axes2)
cla(handles.axes2)
plot(valordeseado1,valorreal1,'-x');
grid on;
title('Error del Instrumento')
xlabel('Valor Experado')
ylabel('Valor Real')

% --- Se ejecuta en cuanto se presiona el Botón Salir.
function pushbutton7_Callback(hObject, eventdata, handles)
ans=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(ans,'No')
return;
end
delete(instrfind('Type', 'serial'));
clear,clc,close all
```

Anexo 3: Código del programa de la Práctica 2.

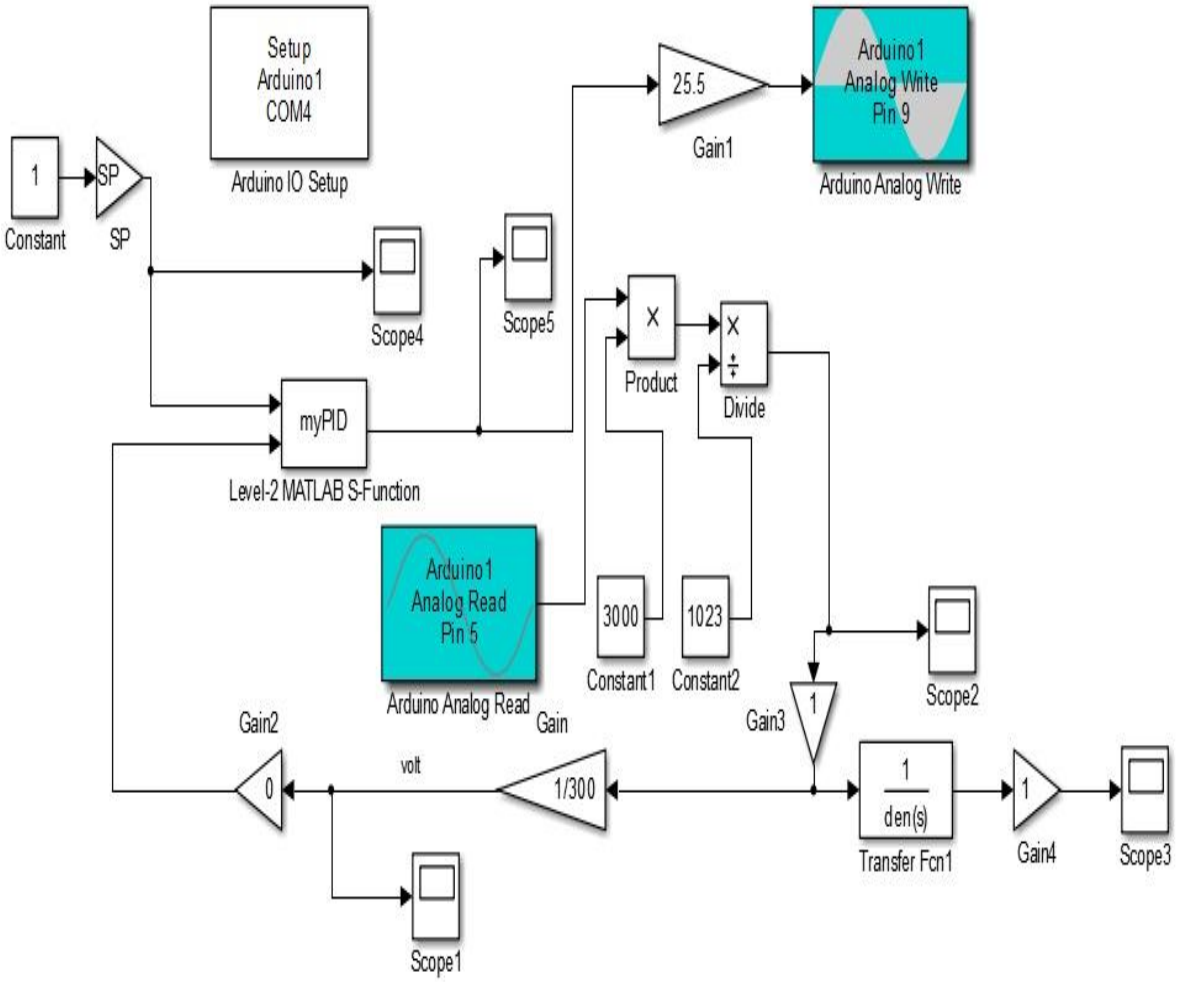
```
% --- Se ejecuta e si selecciona Automático o Manual.
function uipanel1_SelectionChangeFcn(hObject, eventdata, handles)
global Man;
Man=1;
if get(handles.radiobutton1,'value')==1
     Man=1;
else
     Man=0;
end


% --- Se ejecuta en cuanto se presiona el Botón Actualizar Parámetros.
function pushbutton1_Callback(hObject, eventdata, handles)
global Man;
global SP;
global A;
global P;
global I;
global D;
global Tm;
P=0;I=0;D=0;SP=0;A=0;
P=str2double(get(handles.edit2,'string'));
I=str2double(get(handles.edit3,'string'));
D=str2double(get(handles.edit4,'string'));
Tm=str2double(get(handles.edit1,'string'));
Ini=str2double(get(handles.edit9,'string'));
Fin=str2double(get(handles.edit7,'string'));
SP=str2double(get(handles.text7,'string'));
SP = SP/300;
A=str2double(get(handles.text2,'string'));

assignin('base','A',A)
assignin('base','Tm',Tm)
assignin('base','Ini',Ini)
assignin('base','Fin',Fin)
assignin('base','SP',SP)

     find_system('Name','filtra2');
     open_system('filtra2');
     set_param('filtra2/SP','Gain','SP');

if Man==0
    set_param('filtra2/PID Controller','P','1');
    set_param('filtra2/Gain2','Gain','0');
     set_param('filtra2/PID Controller','I','0');
    set_param('filtra2/PID Controller','D','0');
else
    if Man==1
    set_param('filtra2/PID Controller','P',num2str(P));
    set_param('filtra2/PID Controller','I',num2str(I));
    set_param('filtra2/PID Controller','D',num2str(D));
    set_param('filtra2/Gain2','Gain','1');
    end
end
```

```
function eventhandle = localAddEventListener
global var
if var == 0;
eventhandle = add_exec_event_listener('filtra2/Gain3', ...
                                      'PostOutputs',
@localEventListener);
else
eventhandle = add_exec_event_listener('filtra2/Gain4', ...
                                      'PostOutputs',
@localEventListener);
end


    A=evalin('base','A');
    Tm=evalin('base','Tm');
    SP=evalin('base','SP');
    Ini=evalin('base','Ini');
    Fin=evalin('base','Fin');
    set_param('filtra2/SP','Gain','SP');
    set_param('filtra2/Transfer Fcn1','Denominator','([A 1])');

set_param('filtra2','StartTime','Ini');
set_param('filtra2','StopTime','Fin');
set_param('filtra2','FixedStep','Tm');

    save_system('filtra2')

% --- Se ejecuta en cuanto se presiona el Botón Iniciar.
function pushbutton2_Callback(hObject, eventdata, handles)
ModelName = 'filtra2';
open_system(ModelName);                    % Abre el Simulink
set_param(ModelName,'BlockReduction','off');
% Cuando el modelo arranca, llama la funcion localAddEventListener
set_param(ModelName,'StartFcn','localAddEventListener');
% Arranca el modelo
set_param(ModelName, 'SimulationCommand', 'start');

global ph;                          % Crea el manipulador de linea
ph = line([0],[0]);
        grid on;
        title('Velocidad del Motor')
        xlabel('Tiempo(seg)')
        ylabel('Velocidad(rpm)')
find_system('Name','filtra2');

function localEventListener(block, eventdata)

global ph;

%obtiene el valor del tiempo y de la salida del bloque
simTime = block.CurrentTime;
simData = block.OutputPort(1).Data;

%obtiene las coordenadas del punto
xData = get(ph,'XData');
```

```
yData = get(ph,'YData');

%muestra en el axes los últimos n-puntos
n = 3300;

if length(xData) <= n
    xData = [xData simTime];
    yData = [yData simData];
else
    xData = [xData(2:end) simTime];
    yData = [yData(2:end) simData];
end

% Actualiza las coordenadas del punto
set(ph,...
    'XData',xData,...
    'YData',yData);

%tiempo de muestreo del bloque
samplingtime = .01;
offset = samplingtime*n;
xLim = [max(0,simTime-offset) max(offset,simTime)];
drawnow;

% --- Se ejecuta al desplazarce la barra de la constante del Filtro.
function slider1_Callback(hObject, eventdata, handles)
a=get(hObject,'value');
set(handles.text2,'string',num2str(a));

% --- Se ejecuta al marcar en el boton del filtro.
function checkbox1_Callback(hObject, eventdata, handles)
global var;
var=get(hObject,'value');

% --- Se ejecuta al establecer un Set Point en el edit
function edit5_Callback(hObject, eventdata, handles)
var=str2double(get(hObject,'string'));
set(handles.text7,'string',var);
set(handles.slider2,'value',var);

% --- Se ejecuta al desplazarce la barra del Set Point.
function slider2_Callback(hObject, eventdata, handles)
setpoint=get(hObject,'value');
set(handles.text7,'string',num2str(setpoint));
set(handles.edit5,'string',num2str(setpoint));

% --- Se ejecuta en cuanto se presiona el Botón Detener.
function pushbutton3_Callback(hObject, eventdata, handles)
global a;
global sal1;
sal1=0;
ans=questdlg('¿Desea detener la Simulaciòn?','SALIR','Si','No','No');
if strcmp(ans,'No')
return;
end
     find_system('Name','filtra2');
```

```
     open_system('filtra2');
     set_param('filtra2', 'SimulationCommand', 'stop');
     save_system('filtra2')

% --- Se ejecuta al cerrar la aplicación.
function figure1_CloseRequestFcn(hObject, eventdata, handles)
global a;
global sal1;
sal1=0;
ans=questdlg('¿Desea salir del programa?','SALIR','Si','No','No');
if strcmp(ans,'No')
return;
end
     find_system('Name','filtra2');
     open_system('filtra2');
     set_param('filtra2', 'SimulationCommand', 'stop');
     save_system('filtra2')
     close_system('filtra2')
delete(instrfind('Type', 'serial'));
delete(hObject);
```

Anexo 4: Modelo Simulink del control del motor de CD de la Práctica 2.



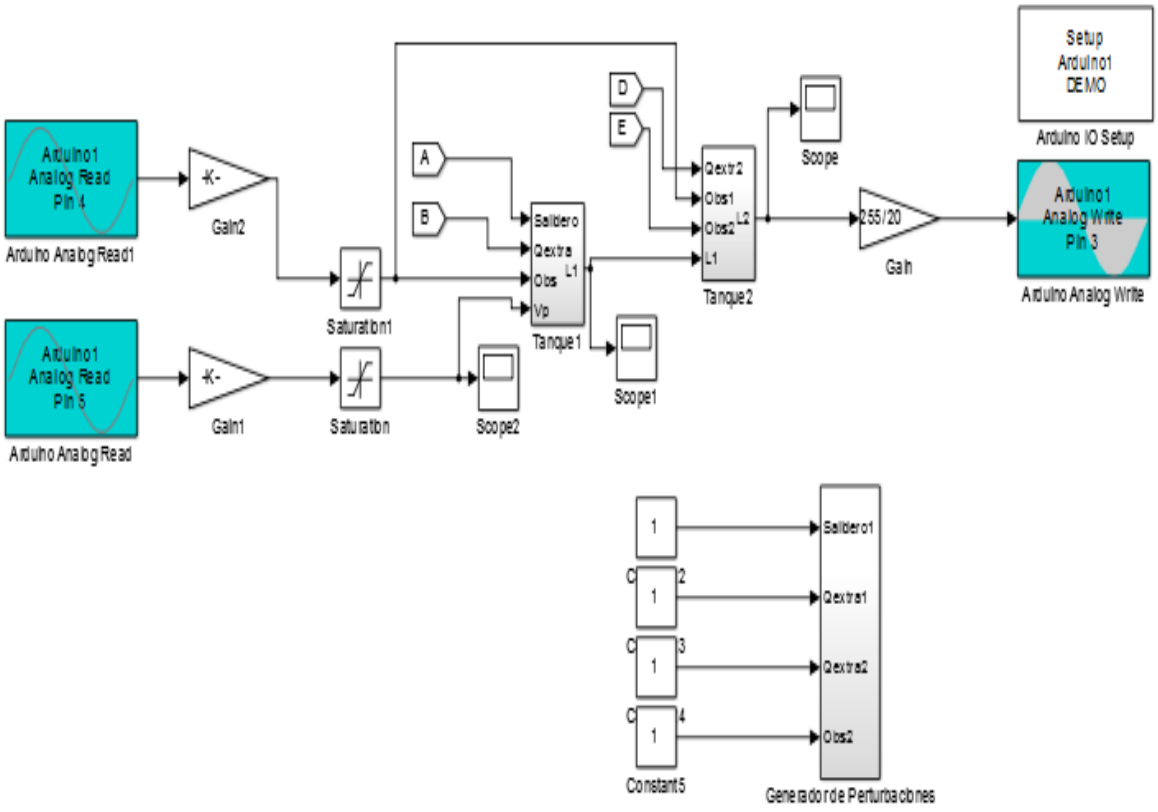
Anexo 5: Código del programa de la Práctica 3.

```
% --- Se ejecuta apenas  se carga el programa.
function emulador_OpeningFcn(hObject, eventdata, handles, varargin)
global nivel1;
global nivel2;
global a;
a=arduino('COM4');
nivel1=25;
nivel2=30;
var=1;

set(handles.activex1,'value',nivel1);
set(handles.activex2,'value',nivel2);

axes(handles.axes1)
background = imread('tank.jpg' );
axis off;
imshow(background);
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
axes(handles.axes4)
background = imread('red.jpg' );
axis off;
imshow(background);
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
axes(handles.axes5)
background = imread('green.jpg' );
axis off;
imshow(background);
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
axes(handles.axes8)
background = imread('red1.jpg' );
axis off;
imshow(background);
%*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
axes(handles.axes9)
background = imread('green1.jpg' );
axis off;
imshow(background);

set (handles.slider1,'value',0);
set (handles.slider2,'value',0);
set(handles.text1,'string',nivel1);
set(handles.text2,'string',nivel2);
set(handles.activex1,'value',nivel1);
set(handles.activex2,'value',nivel2);


% se ejecuta cuando se manipula la variable Obst en tanq 1.
function slider1_Callback(hObject, eventdata, handles)
global nivel1;
global nivel2;
global var
a.pinMode(5,'input');                %% pone el pin 5 en modo lectura
val=str2double(get(handles.text1,'string'));
val1=str2double(get(handles.text2,'string'));
```

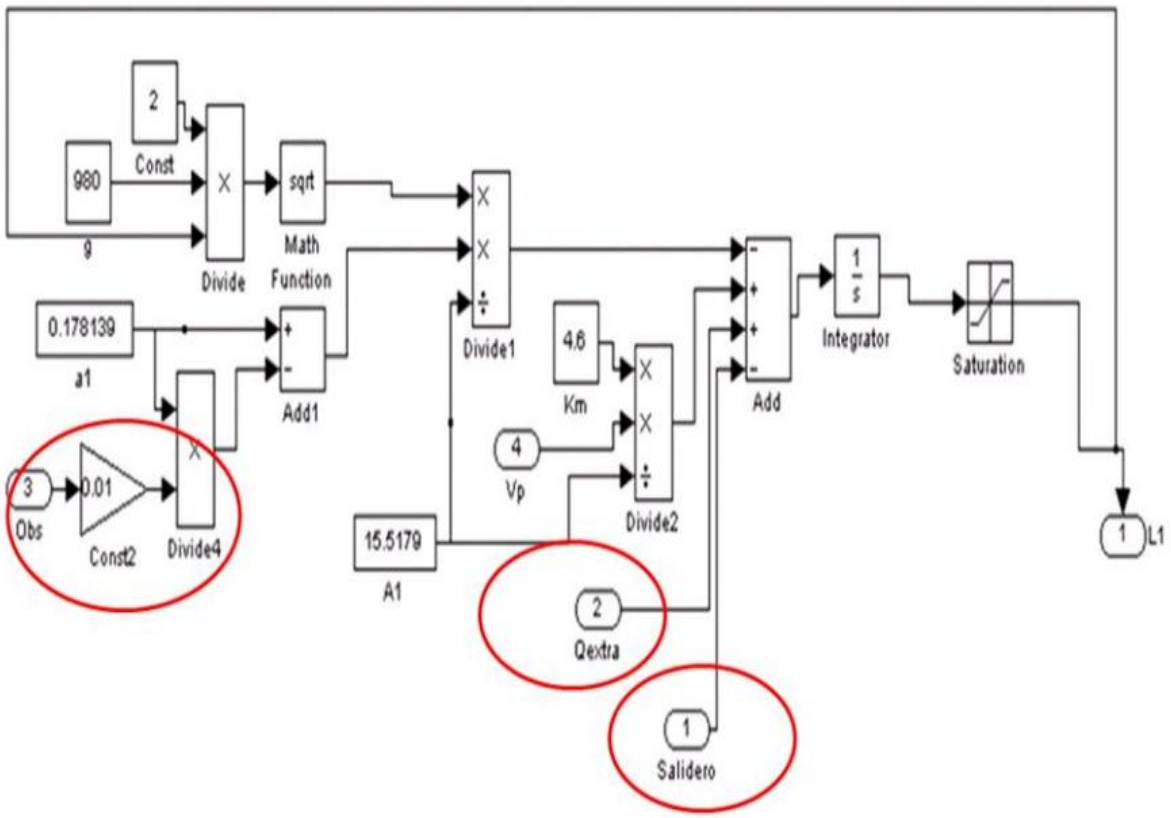
```
obs= a.analogRead(5)
nivel=obs*100/1023;

if (var==nivel)
    var=nivel;
    nivel1=val+1;
    nivel2=val1-1;
    set(handles.activex1,'value',nivel1);
    set(handles.activex2,'value',nivel2);
    set(handles.text1,'string',nivel1);
    set(handles.text2,'string',nivel2);
end
if (var==nivel+1)
    var=nivel;
    nivel1=val-1;
    nivel2=val1+1;
    set(handles.activex1,'value',nivel1);
    set(handles.activex2,'value',nivel2);
    set(handles.text1,'string',nivel1);
    set(handles.text2,'string',nivel2);
    set(handles.activex2,'value',nivel2);
end
if (var==nivel-1)
    var=nivel;
    nivel1=val+1;
    nivel2=val1-1;
    set(handles.activex1,'value',nivel1);
    set(handles.activex2,'value',nivel2);
    set(handles.text1,'string',nivel1);
    set(handles.text2,'string',nivel2);
end



% --- se ejecuta cuando se manipula la variable Flujo de Ent en tanq 1.
function slider2_Callback(hObject, eventdata, handles)
global nivel1;
global a;
global var;

a.pinMode(4,'input');               %% pone el pin 4 en modo lectura
tk1=a.analogRead(4);
nivel=tk1*22/1023;


if (var==nivel)
    var=nivel;
    nivel1=val+1;
    set(handles.activex1,'value',nivel1);
    set(handles.text1,'string',nivel1);
end
if (var==nivel+1)
    var=nivel;
    nivel1=val-1;
    set(handles.activex1,'value',nivel1);
    set(handles.text1,'string',nivel1);
end
```

```
if (var==nivel-1)
    var=nivel;
    nivel1=val+1;
    set(handles.activex1,'value',nivel1);
    set(handles.text1,'string',nivel1);
end


function activex1_Change(hObject, eventdata, handles)
global nivel1;
nivel1=get(hObject,'value');
set(handles.text1,'string',nivel1);


function activex2_Change(hObject, eventdata, handles)
global nivel2;
nivel2=get(hObject,'value');
set(handles.text1,'string',nivel2);


% --- Perturbaciones.
function slider5_Callback(hObject, eventdata, handles)
global nivel2;
tk3=get(hObject,'value');
nivel=tk3*100/1023;
set(handles.text12,'String',nivel);
set(handles.activex1,'value',(nivel2-tk3));


function slider4_Callback(hObject, eventdata, handles)
tk2=get(hObject,'value');
set(handles.text9,'String',(tk2*100/1023));


function slider3_Callback(hObject, eventdata, handles)
global nivel1;
Qextra =get(hObject,'value');
nivel=Qextra*22/1023;
nivel1=nivel1+nivel
```

Anexo 6: Modelo Simulink del proceso de la práctica 3



Anexo 7: Modelo Simulink de las perturbaciones de la práctica 3 (Tanque 1)



Anexo 8: Modelo Simulink de las perturbaciones de la práctica 3 (Tanque 2)