INSTITUTO TECNOLÓGICO SUPERIOR DEL SUR DE GUANAJUATO

Alternativa económica para el monitoreo de planta didáctica de sistema de frenado

Opción 2 Titulación Integral – Tesis profesional

Elaborada por:

Juan Carlos Contreras Guzmán

Que presenta para obtener el título de:

INGENIERO EN SISTEMAS AUTOMOTRICES

Asesor:

Dr. Carlos Alberto Fuentes Hernández

Uriangato, Gto. Noviembre 2022

“Alternativa económica para el monitoreo de planta didáctica de sistema de frenado”

Elaborada por:

Juan Carlos Contreras Guzmán

Aprobado por. . . . . .

Dr. Carlos Alberto Fuentes Hernández
Docente de la carrera de Ingeniería en Electrónica
Asesor de Tesis Profesional

Revisado por. . . . . .

M.C. Mariano Braulio Sánchez
Docente de la carrera de Ingeniería en Sistemas Automotrices
Revisor de Tesis Profesional

Revisado por. . . . . .

M.C. Netzahualcóyotl Martínez Cázares
Docente de la carrera de Ingeniería en Electrónica
Revisor de Tesis Profesional

Instituto Tecnológico Superior del Sur de Guanajuato

TECNOLOGÍA Y CALIDAD PARA LA VIDA

LIBERACIÓN DE PROYECTO PARA LA TITULACIÓN INTEGRAL

Uriangato,Gto. 07/noviembre/2022

Asunto: Liberación de proyecto para la titulación integral

Ing. J. Trinidad Tapia Cruz
Director Académico y de Estudios Profesionales
ITSUR
PRESENTE

Por este medio informo que ha sido liberado el siguiente proyecto para la titulación integral:

Nombre de estudiante y/o egresado(a): Juan Carlos Contreras Guzmán	
Carrera: Ingeniería en sistemas automotrices	Núm. de control: T17120145
Nombre del proyecto: Alternativa económica para el monitoreo de planta didáctica de sistema de frenado	
Producto: Tesis Profesional	

Agradezco de antemano su valioso apoyo en esta importante actividad para la formación profesional de nuestras y nuestros egresados.

ATENTAMENTE

M.C Mariano Braulio Sánchez
Coordinador de Ingeniería en Sistemas Automotrices
ITSUR

La comisión revisora ha tenido a bien aprobar la reproducción de este trabajo.

Dr. Carlos Alberto Fuentes Hernández. Asesor de la Tesis Profesional	M.C Mariano Braulio Sánchez Revisor de la Tesis Profesional	Ing. Netzahualcóyotl Martínez Cazares Revisor de la Tesis Profesional

c.c.p.- Expediente

Instituto Tecnológico Superior del Sur de Guanajuato
División de Ingeniería en Sistemas Automotrices

DEPARTAMENTO ACADEMICO	CLAVE:11EIT0002E
	ISA-EGR-2022/27

Uriangato, Guanajuato, 07/noviembre/2022

Asunto: Aprobación de impresión de trabajo profesional

C. JUAN CARLOS CONTRERAS GUZMÁN
P R E S E N T E:

Por medio de este conducto, le comunico a usted que después de haber sido revisado su trabajo bajo la cual se derivó la Monografía Titulada:

"Alternativa económica para el monitoreo de planta didáctica de sistema de frenado"

La comisión revisora, ha tenido a bien aprobar la impresión de este trabajo.

A T E N T A M E N T E
"Excelencia en Educación Tecnológica"®
"Tecnología y Calidad para la Vida"

Instituto Tecnológico Superior del Sur de Guanajuato
COORDINACIÓN INGENIERÍA EN SISTEMAS AUTOMOTRICES

M.C Mariano Braulio Sánchez
Jefe de División de Ingeniería en Sistemas Automotrices

C.c.p Unidad de Servicios Escolares
C.c.p Coordinación de Ingeniería en Sistemas Automotrices
C.C.p Archivo Consecutivo

Agradecimientos

1. A Dios por haberme por haberme puesto a las personas correctas que estuvieron para mí con su apoyo y motivación para que todo esto haya sido posible.
2. A mi asesor el Dr. Carlos Alberto Fuentes por darme la oportunidad de trabajar en este proyecto y la asesoría brindada durante la realización del mismo.
3. Al encargado de laboratorios C. Cuauhtémoc Duran por las facilidades para hacer uso de los materiales e instalaciones para la realización del proyecto.
4. Al coordinador de la carrera ingeniería en sistemas automotrices M.C. Mariano Braulio Sánchez por su ayuda en el proceso de titulación.
5. Y a cada uno de mis profesores que tuve a lo largo de la carrera que me enseñaron las bases de conocimientos que fueron base para la realización de este proyecto.

Dedicatoria

Este trabajo va dedicado a mis padres ya que sin ellos nada de esto hubiera sido posible, gracias a ellos soy la persona que soy y gracias a ellos pude cumplir mi meta de estudiar esta carrera, de igual modo va dedicado para mis abuelos que desde el cielo sé que me cuidan y aunque físicamente ya no estarán para verme graduar. Sé que desde el cielo se alegraran al verme terminar mi carrera como ellos deseaban verme.

De igual manera se lo quiero dedicar a mí madrina Marisol Escobedo por su gran ayuda y apoyo a lo largo de estos ya casi cinco años, y también es para mis amigos de la universidad Jesús Zamudio y Erick García que se convirtieron en casi una familia para mí, gracias a ellos por siempre motivarme y ayudarme a estudiar cuando los temas eran difíciles para mí. Y de igual manera para cada una de las personas que fueron parte de este capítulo de mi vida llamado universidad.

Título de la tesis:

Resumen:

Este proyecto surgió como continuación a lo realizado anteriormente en el proyecto de instrumentación de planta didáctica del sistema de frenado, con la finalidad de mejorar lo ya hecho, en el caso del presente proyecto se realizó la medición de velocidad ahora si con un sensor y no solo con una emulación de la señal del mismo, se hicieron adaptaciones y mejoras al programa, se realizó la comparación entre los valores obtenidos en el VI y un tacómetro digital para comprobar su eficacia al momento de medir las RPM, además de logro obtener la curva de desaceleración natural de la planta y su comparación con un modelo matemático que se asemeja al comportamiento de la planta de frenado.

Palabras claves

Arduino, LabVIEW, PWM, RPM, Comunicación serial, velocidad, desaceleración, temperatura.

Abstract

This project arose as a continuation of what was previously done in the project of instrumentation of the didactic plant of the braking system, with the purpose of improving what has already been done, in the case of the present project the speed measurement was carried out now if with a sensor and not only with an emulation of the signal of the same, adaptations and improvements were made to the program, the comparison between the values obtained in the VI and a digital tachometer was made to verify its effectiveness when measuring the RPM, in addition to obtaining the curve natural deceleration of the plant and its comparison with a mathematical model that resembles the behavior of the braking plant.

Índice de tablas

Índice de figuras

Tabla de contenido

Capítulo 1
Introducción.

Uno de los sistemas de mayor importancia para la seguridad de un automóvil y sus tripulantes es el sistema de frenado, el conocimiento sobre dicho sistema se convierte de vital importancia para un ingeniero en sistemas automotrices, es por ello que una planta didáctica del sistema de frenado en una institución como el ITSUR debe ser de gran ayuda para la formación de sus alumnos. Para que los alumnos comprendan bien el funcionamiento del sistema es mejor que lo vean de manera física en lugar de que solamente se conformen con lo que leen de teoría dentro del aula de clase.

Es por ello que se está trabajando para lograr una planta completamente funcional, lo desarrollado en este proyecto es una aportación para conocer los parámetros importantes del funcionamiento de la misma como lo son la velocidad de giro de la llanta y la temperatura en las balatas que forman parte del sistema mecánico.

Este proyecto es una continuación a lo realizado anteriormente para poder mejorar lo ya hecho y poder tener una mejor funcionalidad del sistema, en el caso de este proyecto se hace uso de dos instrumentos virtuales (VI) de LabVIEW uno para velocidad y temperatura, y otro para la activación de la válvula que emula un ABS, todo esto con la finalidad de no exigir tanto a una placa Arduino y que esto conlleve al mal funcionamiento del mismo o incluso a la detención del programa. Este proyecto también hace uso de la comunicación serial entre la placa Arduino y un VI de LabVIEW. La principal diferencia con algún proyecto anterior es el hecho de que en este caso ya se hizo uso de un sensor para detectar la velocidad real de la llanta y no de un generador de funciones como antes se emulaba. Para el caso de la válvula, al aún no estar colocada en la planta su posible funcionamiento fue obtenido

mediante el osciloscopio para poder observar la modulación PWM correspondiente al valor seleccionado.

Otra aportación es el hecho de poder comparar las mediciones del VI con las de un tacómetro digital para corroborar las mediciones obtenidas y comprobar que no estén tan alejadas de la medida que se obtiene con un aparato que es diseñado específicamente para medir las RPM. En este trabajo, con los datos obtenidos se pudo obtener la curva de desaceleración que presenta la planta de manera natural sin la intervención del sistema mecánico, misma que presenta datos que ayudan a la mejor interpretación del sistema, esto gracias a las herramientas que se desarrollan en LabVIEW, en este proyecto solo la obtención de los datos fueron realizados con la placa Arduino uno, de ahí fueron enviados para su manipulación y conversión a los datos de interés a un VI, esto con la finalidad de no consumir los recursos de la placa y no exigir tanto a la misma dadas sus limitaciones.

Este proyecto aporta la instrumentación para la planta del sistema de frenado lo cual ya es de ayuda para el funcionamiento de la misma y solo bastaría la colocación de la válvula para poder tener la planta al 100%.

A continuación, se presenta lo realizado en este proyecto.

Capítulo 2

Marco teórico (Antecedentes).

Plantas didácticas de frenado en el mercado

En el mercado actual existen algunos tipos de plantas didácticas algunas de las existentes se mencionan a continuación.

MTF equipo de freno de tambor

El Equipo de Freno de Tambor, "MFT", permite el estudio de las fuerzas que actúan en un freno de tambor, la diferencia entre el par de frenado, entre los sistemas de frenado de la zapata primaria y secundaria y determinar el coeficiente de fricción entre el tambor y la zapata de freno, además con el equipo se pueden estudiar los distintos tipos de configuraciones de zapatas existentes, la figura 1 ilustra el equipo antes mencionado, marca Edibon, modelo MTF, cuyo costo suele empezar en los cuatro mil euros (Edibon, 2022).

Fuente Edibon

Figura 1. MTF equipo de freno de tambor.

Sistema de frenado DL AM11

Este panel está compuesto por un freno de disco en la llanta delantera y por un freno de tambor en la llanta motriz. Ambas ruedas rotan lentamente. Cuando el freno es activado, ambas llantas se bloquean. El cilindro se mueve hidráulicamente. El sistema cubre los siguientes argumentos:

- Llanta posterior blocada, la presión no disminuye al soltar el pedal
- Pérdida de vacío
- Falla en el freno posterior
- Falla en el freno delantero
- Freno de mano
- Falla en la luz de stop

El panel está completo de CAI software. A continuación, se ilustra el sistema de frenado en la figura número 2 siguiente, marca De Lorenzo, modelo DL AM11, (De Lorenzo, 2022).

Fuente De Lorenzo

Figura 2. Sistema de faenado hidráulico DL AM11.

Planta didáctica para el entrenamiento de sistemas automotrices

Proyecto elaborado por los alumnos: Manuel Alejandro Pérez Martínez, Oscar Servín Loa, Heriberto López Raya, asesorados por el Dr. Carlos Alberto Fuentes Hernández. Ese proyecto es la base para el presente, en dicho proyecto se realizó la fabricación de la estructura física de la planta (Manuel Alejandro Pérez Martínez, 2019)), su imagen se muestra en la figura 3, para su funcionamiento inicial se hizo uso de un PLC y para la manipulación del mismo se utilizó un HMI. En ese proyecto la velocidad de la llanta se midió con tacómetros manuales, mediante el HMI se prefijaba la velocidad del motor y se lograba la activación del freno. En LabVIEW se inició el planteamiento del monitoreo de los parámetros de velocidad y temperatura, pero se avanzó poco. Luego, la adquisición de datos se migró hacia una tarjeta MyRio para detectar los flancos de subida que serían utilizados para logar calcular la velocidad de giro de la rueda, quedando el prototipo sin grandes avances por el aumento de su costo.

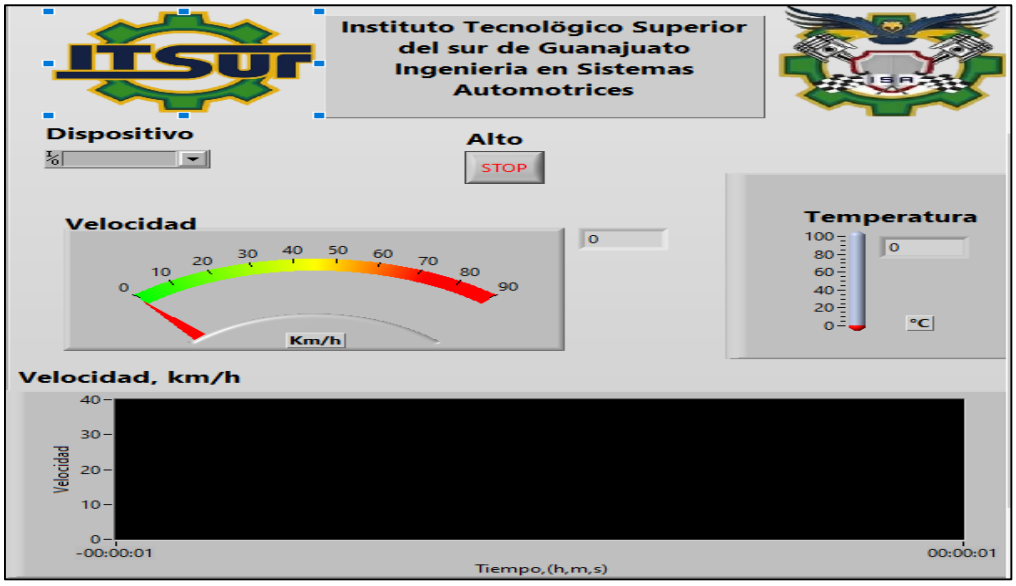
Fuente Pistas educativas

Figura 3. Prototipo planta didáctica.

Instrumentación de planta didáctica del sistema de frenado.

Proyecto realizado por Juan Carlos Contreras Guzmán bajo la supervisión del Dr. Carlos Alberto Fuentes Hernández en las instalaciones del Instituto Tecnológico Superior del sur de Guanajuato, el cual consistió en determinar la viabilidad de la adquisición de los datos a bajo costo, utilizando una placa Arduino uno y su comunicación vía puerto serial con LabVIEW para el monitoreo de las condiciones de funcionamiento durante el frenado. Por las limitaciones en el tiempo para la realización de este proyecto, no se instaló nada sobre la planta y los resultados experimentales se obtuvieron con instrumentos de banco (generador y osciloscopio), la velocidad de la llanta se emuló con un tren de pulsos en lugar de un sensor En este proyecto se implementó además el control de la válvula dentro del mismo VI de LabVIEW. (Guzmán, 2022).

De este proyecto surgió este trabajo de tesis para mejorar lo ya hecho y en este caso ya hacerlo de manera física con los sensores adecuados, de tal manera que ahora los resultados sean reales y no simulados. El VI de este proyecto es el que se muestra a continuación.

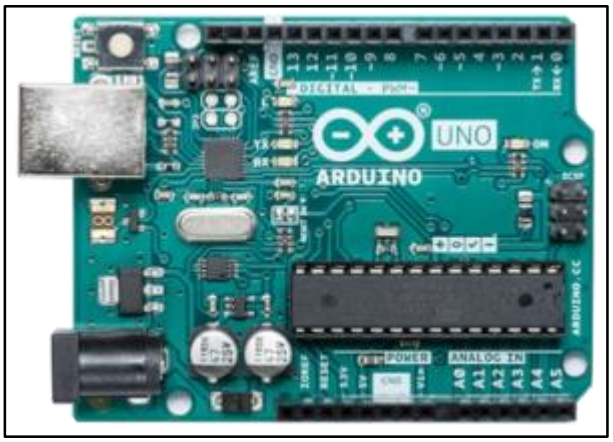


Fuente creación propia

Figura 4. VI Proyecto base.

Arduino uno

Arduino es una plataforma de desarrollo basada en una placa electrónica de hardware libre que incorpora un microcontrolador re-programable y una serie de pines hembra. Estos permiten establecer conexiones entre el microcontrolador y los diferentes sensores y actuadores de una manera muy sencilla (principalmente con cables DuPont), su modelo más utilizado es el Arduino uno mismo que se muestra en la figura número 5.



Fuente Arduino

Figura 5. Placa Arduino uno.

Una placa electrónica es una PCB (“Printed Circuit Board”, “Placa de Circuito Impreso” en español). Las PCBs superficies planas fabricadas en un material no conductor, la cual consta de distintas capas de material conductor. Una PCB es la forma más compacta y estable de construir un circuito electrónico. Por lo tanto, la placa Arduino no es más que una PCB que implementa un determinado diseño de circuitería interna. De esta forma el usuario final no se debe preocupar por las conexiones eléctricas que necesita el microcontrolador para funcionar, y puede

empezar directamente a desarrollar las diferentes aplicaciones electrónicas que necesite.

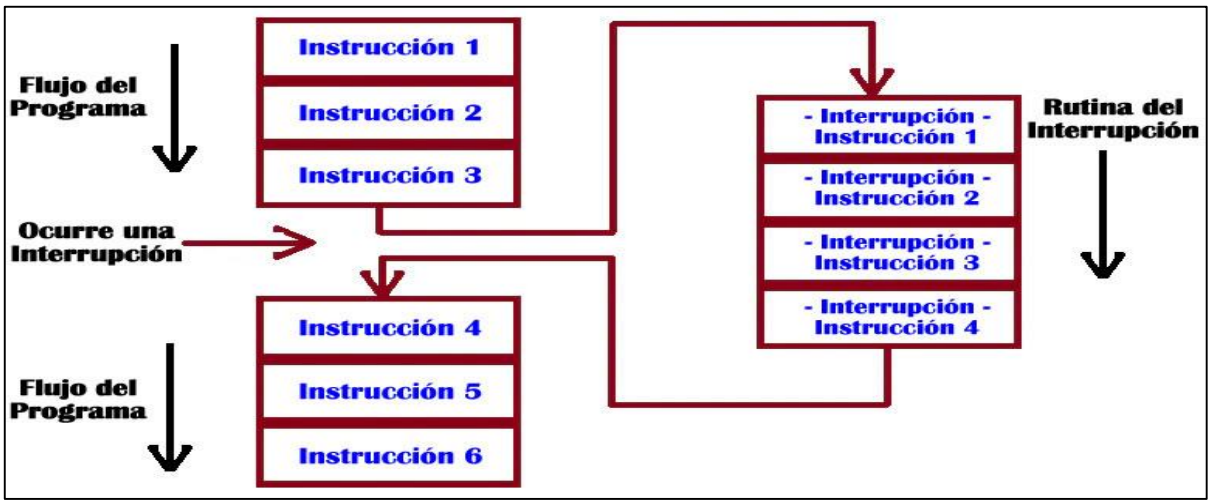
El microcontrolador que lleva la placa Arduino UNO es el modelo ATmega328P de la marca Atmel. La “P” del final significa que este chip incorpora la tecnología “Picopower” (propietaria de Atmel) y permite un consumo eléctrico ligeramente menor comparándolo con el modelo equivalente sin “Picopower”, ATmega328 (sin la “P”) (Arduino, 2022).

Características placa Arduino uno

- Microcontrolador: ATMega328P.
- Velocidad de reloj: 16 MHz.
- Voltaje de trabajo: 5V.
- Voltaje de entrada: 7,5 a 12 voltios.
- Pinout: 14 pines digitales (6 PWM) y 6 pines analógicos.
- 1 puerto serie por hardware.
- Memoria: 32 KB Flash (0,5 para bootloader), 2KB RAM y 1KB Eeprom (Arduino, 2022).

Interrupciones en Arduino

Una interrupción es una señal que indica al procesador cuándo ocurre un evento que requiere atención inmediata. El procesador debe responder a esta señal interrumpiendo las instrucciones actuales y pasando el control al controlador de interrupciones (ISR, Interrupt Service Routine). El controlador es una función común que escribimos nosotros mismos y colocamos allí el código que debe responder al evento. El funcionamiento sería el que a continuación se ilustra en la figura 6.



Fuente Proyectos con Arduino

Figura 6. Esquema de funcionamiento de las interrupciones.

Después de la interrupción del servicio ISR, la función termina y el procesador continúa con las instrucciones interrumpidas – continúa ejecutando el código desde el punto donde fue detenido. Todo esto sucede automáticamente (Proyectos con Arduino, 2022). A continuación, en la tabla número 1 se muestra los pines de interrupciones en cada una de las placas de la familia Arduino.

Tabla 1. Pines de interrupciones familia de placas Arduino.

PLACA	PINES PARA INTERRUPCIONES
UNO, NANO, MINI	2,3
MEGA, MEGA2560, MEGAADK	2,3,18,19,20,21
MICRO, LEONARDO	0,1,2,3,7
ZERO	Todos los pines digitales excepto el 4
MKR1000 REV.1	0,1,4,5,6,7,8,9, A1,A2
DUE	Todos los pines digitales

Fuente Proyectos con Arduino

La función attachInterrupt se utiliza para manejar las interrupciones. Se usa para conectar una interrupción externa al controlador.

Sintaxis de la llamada: attachInterrupt (Interrupt, function, mode)

Argumentos de la función:

- interrupt – número de la interrupción llamada (estándar 0 – para el 2º pin, para la placa Arduino Uno 1 – para el 3º pin),
- function – el nombre de la función que se llamará en la interrupción (es importante que la función no acepte ni devuelva ningún valor),
- mode – es una condición para activar una interrupción.

Se pueden ajustar las siguientes condiciones de funcionamiento:

- LOW – se realiza en un nivel de señal bajo cuando el contacto es cero. La interrupción puede repetirse cíclicamente, por ejemplo, cuando se pulsa el botón.
- CHANGE – la interrupción ocurre cuando la señal cambia de alta a baja o viceversa. Se realiza una vez en cualquier cambio de señal.
- RISING – interrumpe una vez al cambiar la señal de LOW a HIGH.
- FALLING – interrumpe una vez al cambiar la señal de HIGH a LOW (Proyectos con Arduino, 2022).

Comunicación serial

La comunicación serial se suele emplear para comunicar Arduino con un ordenador u otros dispositivos. Todas las placas Arduino tienen al menos un puerto serie también nombrado como UART el cual comunica los pines digitales 0(RX) y 1(TX) con el ordenador a través de USB. UART significa “Universal Asynchronous Receiver-Transmitter” y es un controlador de puertos y dispositivos en serie. Normalmente se encuentra integrado en la placa base y sirve para manejar las interrupciones de los dispositivos conectados al puerto serie y convertir los datos en formato paralelo en formato serie para que puedan ser transmitidos a través de los puertos y viceversa. Los UART son programables y debe configurarse la velocidad, la paridad, la longitud y los bits de parada. El UART viene incluido en el hardware de Arduino integrado en el chip, el cual permite la comunicación incluso mientras se trabaja en otras tareas, siempre que haya un espacio en el buffer de serie de 64 bytes.

El puerto es el nombre genérico para definir las interfaces, físicos o virtuales, que permiten la comunicación entre dos ordenadores o dispositivos. El puerto serie (o puerto serial) es una interfaz de comunicaciones de datos digitales donde la información es transmitida bit a bit, es decir, envía un único bit a la vez en lugar de varios bits simultáneos como ocurre con la transmisión en paralelo que envía varios bits simultáneamente. Esta secuencia de bits es la forma de enviar la información a través de los dispositivos.

La transferencia de datos a través de los puertos de serie se emplea con frecuencia asociándose sobre todo al estándar RS-232; no obstante, últimamente se ha optado por la comunicación USB la cual manda los datos como un flujo en serie debido a que es más rápida la transmisión (Hidalgo, 2022).

LabVIEW

LabVIEW es una plataforma de software para programación que proporciona un potente entorno de desarrollo gráfico para el diseño de aplicaciones de Ingeniería de adquisición de datos, análisis de medidas y presentación de datos gracias a un lenguaje de programación sin la complejidad de otras herramientas de desarrollo.

Características principales:

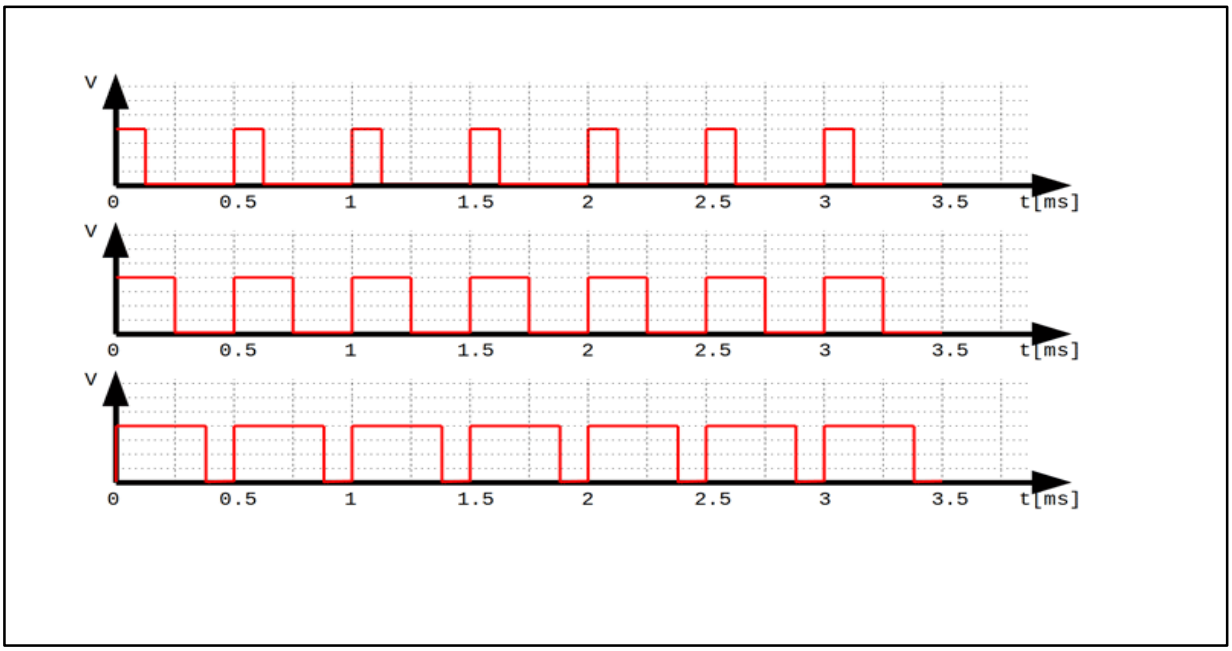
1. Intuitivo lenguaje de programación.
2. Herramientas de desarrollo y librerías de alto nivel específicas para aplicaciones.
3. Cientos de funciones para E/S, control, análisis y presentación de datos.
4. Posibilidad de crear aplicaciones de medida genéricas sin programación.

5. Depuración gráfica integrada y control del código fuente.
6. Miles de programas de ejemplo, tanto en el software como por web.
7. Ayuda contextual integrada y extensos tutoriales (Universidad de Cantabria, 2016).

Modulación PWM

El término de señal PWM proviene del inglés ***Pulse Width Modulation*** que significa modulación por ancho de pulsos. Una señal PWM es una señal digital similar al tren de pulsos cuadrado.

La principal diferencia con el tren de pulsos es que en la señal PWM, es posible variar el tiempo que la señal se mantiene en estado alto, pero siempre manteniendo el periodo constante, tal y como se muestra en la figura 7 de a continuación.



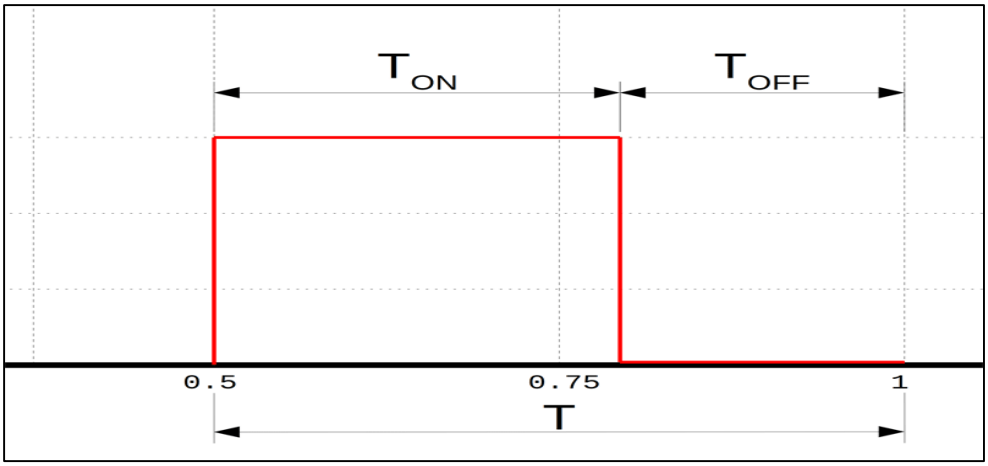
Fuente programa fácil

Figura 7. Tren de pulsos.

Esta capacidad de variar el tiempo en estado alto, es lo que realmente hace que la señal PWM sea tan útil y práctica:

- Pueden utilizarse para controlar un servomotor mediante señales pulsantes con diferentes tiempos en alto.
- Sirven para emular una salida analógica.
- Es muy empleada en algunas fuentes de alimentación en la etapa de regulación.

Dos conceptos importantes en la señal PWM son el tiempo de encendido y tiempo apagado mismos que se ilustran a continuación en la figura 8.



Fuente programa fácil

Figura 8. Tiempo de encendido y tiempo de apagado.

En la imagen se han destacado tres valores de tiempo:

- T_{ON}: tiempo que la señal se mantiene en estado alto
- T_{OFF}: tiempo que la señal se mantiene en estado bajo
- T: periodo de la señal.

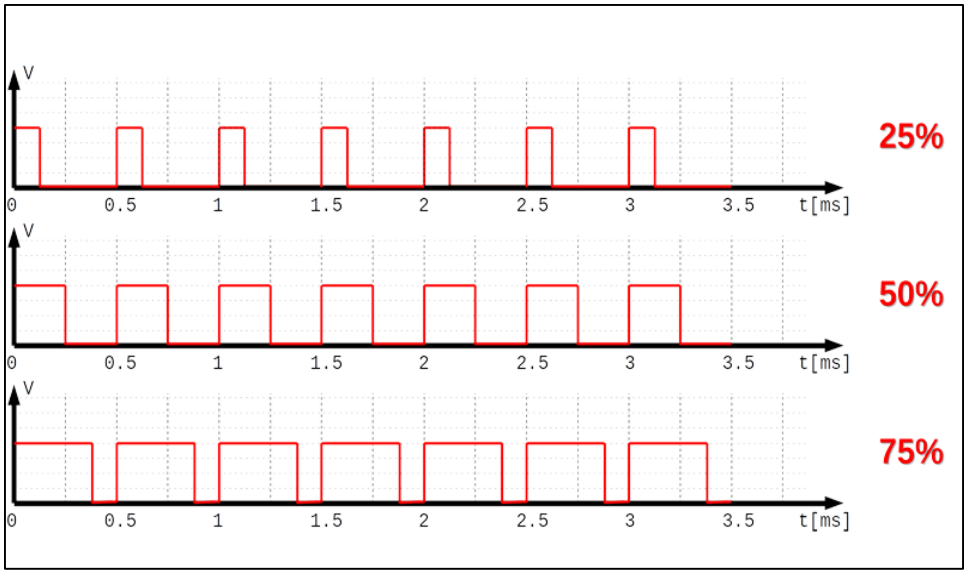
En cualquier caso, siempre se cumple: T = T_{ON} + T_{OFF}. Es lógico, el tiempo total (T) es igual al tiempo que está en estado alto (T_{ON}) más el tiempo que está en estado bajo (T_{OFF}).

El ciclo de trabajo o duty cycle de la señal es uno de los conceptos más importantes de una señal PWM. Este se representa mediante la letra D y se define como la razón entre el tiempo en estado alto (T_{ON}) y el periodo de la señal (T), se puede calcular con la formula siguiente:

$$D = \frac{T_{on}}{T} * 100\%$$

(Ec 1)

Por ejemplo, un ciclo de trabajo igual a 50% quiere decir que la señal está la mitad del tiempo en estado alto y la otra mitad en estado bajo. Si es al 25% estará el 25% del tiempo en estado alto y el 75% en estado bajo. Algunos ejemplos de señales se muestran en la figura 9 de a continuación.



Fuente programa fácil

Figura 9. Ejemplos de señales PWM.

Tabla 2. Pines para modulación PWM.

Placa	*Pines*	*Frecuencia*
Uno, nano, mini	3, 5,6,9,10,11	490 Hz (pines 5 y 6: 980 Hz)
Mega	2-13, 44-46	490 Hz (pines 4 y 13: 980 Hz)
Leonardo	3,6,9,10,11,13	490 Hz (pines 3 y 11: 980 Hz)
Uno Wifi Rev2	3, 5, 6, 9, 10	976 Hz
Nano 33	1-13, A0-A7	500 Hz
NodeMCU Esp8266	1-4, 5-8, 12	1 kHz

Fuente programa fácil

La tabla 2 anterior muestra los diferentes pines de PWM, así como la frecuencia que se maneja en cada uno de ellos dentro de las diferentes placas más comunes en el mercado.

Hardware

Sensor de temperatura LM335

Son sensores de temperatura de circuito integrado de precisión y fácilmente calibrado. Al operar como un Zener de 2 terminales, el LM335 tiene una tensión de ruptura directamente proporcional a la temperatura absoluta a 10 mV/ ° K. Con una impedancia dinámica inferior a 1 Ω, el dispositivo funciona en un rango de corriente de 400 µA a 5 mA con prácticamente ningún cambio en el rendimiento. Calibrado a 25 °C, el LM335 tiene un error inferior a 1 °C en un rango de temperatura de 100 °C. A diferencia de otros sensores, el LM335 tiene una salida lineal.

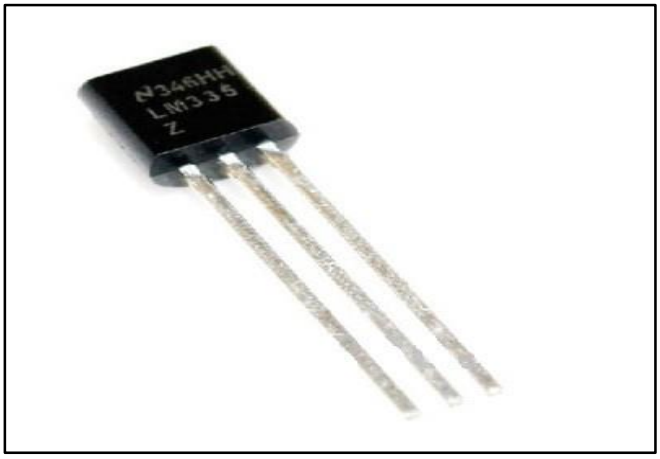
Las aplicaciones para el LM335 incluyen casi cualquier tipo de sensor sobre un rango de temperatura de – 40 °C a 100 °C. La baja impedancia y la salida lineal permiten que la interconexión para la lectura o control del circuito sea especialmente fácil (Digi-Key, 2022).

Características

- Calibrado directamente a la escala de temperatura Kelvin
- 1 º C de precisión inicial disponible
- Opera desde 400 µA a 5 mA
- Impedancia dinámica inferior a 1 ΩFácilmente calibrado

- Amplio rango de temperatura de funcionamiento
- Rango amplio de 200 °C
- Bajo costo

La figura número 10 ilustra un ejemplo de un sensor de estas características.



Fuente Digi-Key

Figura 10. Sensor LM335

Sensor Banner QS30E

Los sensores fotoeléctricos, también conocidos como células fotoeléctricas, emiten un rayo de luz que se usa para detectar la presencia o ausencia de elementos y equipos o cambios en las condiciones de una superficie. Cuando la luz emitida es

interrumpida o reflejada por el objeto, el cambio en los patrones de luz es medido por un receptor y el objeto o superficie es reconocido. Los sensores fotoeléctricos (figura 11) son muy comunes en áreas de fabricación industrial, como manipulación de materiales, embalaje, alimentos y bebidas, medicina y muchos otros.

Dependiendo del estilo seleccionado, se pueden usar con o sin un reflector, o ser autónomo, de largo alcance, resistente o compacto. Hay muchas opciones diferentes de alojamiento y montaje para ofrecer un ajuste correcto que satisfaga las demandas de cada aplicación (Banner, 2022).

Fuente t.ly/gZd0

Figura 11. Sensor Banner QS30E.

LM311P

Los dispositivos LM111, LM211 y LM311 son solo comparadores de voltaje de alta velocidad. Estos dispositivos son diseñados para operar desde una amplia gama de voltajes de fuente de alimentación, incluidos suministros de ± 15 V para

amplificadores operacionales y suministros de 5 V para lógica sistemas Los niveles de salida son compatibles con la mayoría Circuitos TTL y MOS. Estos comparadores son capaz de encender lámparas o relés y conmutar voltajes hasta 50 V a 50 mA. Todas las entradas y salidas pueden aislarse de la tierra del sistema. Las salidas pueden cargas de accionamiento referenciadas a tierra, VCC+ o VCC−. Compensar las capacidades de equilibrio y luz estroboscópica está disponibles, y las salidas se pueden conectar por cable-O. Si la luz estroboscópica es bajo, la salida está en el estado de apagado, independientemente de la entrada diferencial (Texas instruments, 2022).

Sus encapsulados suelen ser como el de la figura 12.

Fuente t.ly/4b_V

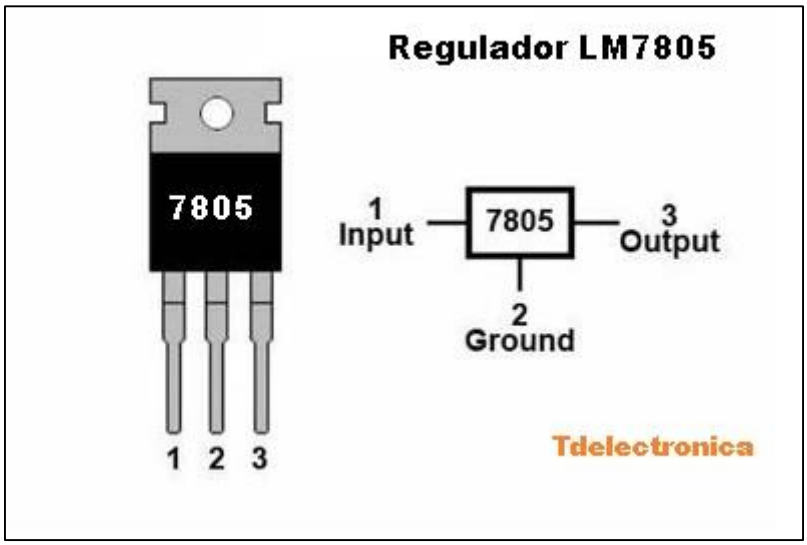
Figura 12. LM311P.

Regulador LM7805

El LM7805 es un regulador de voltaje con salida fija de 5 volts. Es muy popular debido a su bajo costo y facilidad de uso, ya que solamente requiere un par de capacitores externos para funcionar. El regulador pertenece a la familia 78XX que incluye circuitos integrados reguladores con salidas fijas en diferentes voltajes.

La principal función del LM7805 es mantener en su salida un voltaje estable, a pesar de las variaciones que reciba en la entrada. La resistencia interna del regulador varía dependiendo del voltaje de entrada y de la carga, dando como resultado una salida de voltaje constante. Se puede decir entonces que un regulador de tipo lineal actúa como un divisor de voltaje que se ajusta de forma automática constantemente para mantener la salida de voltaje fija (Geek factory, 2022).

Físicamente son como se muestra en la figura 13 que a continuación se presenta.



Fuente Texas instruments

Figura 13. Regulador LM7805

Capítulo 3

Planteamiento del problema

3.1. Identificación. Dentro del Instituto Tecnológico Superior del Sur de Guanajuato la carrera de ingeniería en sistemas automotrices es una de las más recientes, es por ello que se busca incrementar los materiales didácticos en laboratorio para instrucción de los alumnos, de esto surge la propuesta de la planta didáctica del sistema de frenado automotriz.

3.2. Justificación. La realización de esta planta será de gran ayuda para los estudiantes de la carrera en sistemas automotrices y otras más que deseen apoyarse en ella para sus materias, en el caso de sistemas automotrices seria en gran ayuda para materias como elementos automotrices y dinámica, actualmente en el instituto no existe algo similar por lo que sería una gran innovación y aportación para el laboratorio de ingeniería en sistemas automotrices.

3.3. Alcance. El presente desarrollo tiene como finalidad monitorear la velocidad de giro de la rueda, conocer la temperatura de la balata, así como activar la válvula de frenado en forma pulsada, todo esto mediante un instrumento construido en LabVIEW, conectado por el puerto serial a una placa Arduino. Lo que no se abordará es utilizar otra llanta para que mediante la válvula del ABS igualar la velocidad de ambas para que no exista el bloqueo de alguna de ellas y puedan frenar al mismo tiempo.

Capítulo 4

Objetivos

4.1. Objetivo general. Desarrollar la instrumentación que realice la medición de velocidad de giro de una rueda, active o desactive los frenos en forma pulsada y mida su temperatura, todo esto con hardware de bajo costo.

4.2. Objetivos específicos.

- Monitorear velocidad de la rueda y temperatura de balata.
- Programar en Arduino y LabVIEW.
- Establecer comunicación serial entre placa Arduino y LabVIEW.
- Visualización de parámetros en un VI de LabVIEW.
- Obtener curva de desaceleración natural de la planta.
- Activar válvula por medio de un VI.

Capítulo 5

Metodología

Este proyecto fue realizado en las instalaciones del Instituto Tecnológico Superior del Sur de Guanajuato, en el laboratorio de electrónica, que es el lugar donde está colocada la planta de frenado, para un mejor entendimiento de lo realizado será descrito lo hecho en secciones para su entendimiento.

Adquisición de flancos

Para conocer la velocidad a la cual gira la rueda se hace uso de la placa Arduino uno, es por ello que la programación se realiza en la plataforma de Arduino IDE, teniendo como primer paso declarar la variable que actuará como el contador de los flancos, en este caso se denomina como de tipo entero y bajo el nombre de NF, lo siguiente es asignar el pin por el cual se detectaran los flancos, por tratarse de una placa Arduino UNO los pines solo pueden ser el número dos o el tres, en este caso se seleccionó el dos, que también es de tipo entero y con el nombre de detector, realizado lo anterior ahora se procede a utilizar una de las funciones de Arduino, que es la función de interrupciones, la cual es utilizada para detectar los flancos, la sintaxis de la misma fue descrita en la sección de marco teórico, para el caso de este proyecto dicha sintaxis es como se muestra en la siguiente figura 14. Además se declara la velocidad de comunicación serial que en este caso es de 9600.

```
attachInterrupt(digitalPinToInterrupt(detector),contador,FALLING);
Serial.begin(9600); // put your setup code here, to run once:
```

Fuente creación propia.

Figura 14. Función de interrupción.

En este caso, cuando se detecte un flanco por medio del pin asignado el programa pasa a la función del contador en donde ya se tiene asignado que se incremente el valor de NF (ver figura 15), y por último la instrucción de FALLING para indicar que se utilizan flancos de bajada.

```
void contador ()
{ NF++;
}
```

Fuente creación propia

Figura 15. Void contador.

Al tener un contador también es necesario reiniciar el mismo, es por ello que en este caso se hace uso de la función de cociente y residuo para tener un valor numérico que pueda reiniciar el contador en una determinada situación, en este caso el valor a utilizar para dicha función es el tiempo el cual se encuentra en milisegundos, se utilizan otras dos variables de tipo doublé mismas que se declaran al inicio del programa una con el nombre de cociente y la otra como residuo, el valor de interés es el residuo el cual siempre será un valor entre 1 y 1000, dicho valor se calcula de la manera que se ilustra en la figura 16 de a continuación.

```
tiempo = millis();
cociente = (tiempo/1000);
residuo = tiempo -(cociente*1000);
```

Fuente creación propia

Figura 16. Función cociente residuo.

Una vez con el valor del residuo se crea una estructura IF para que dado un valor 980 en el residuo se restablezca el contador, se determinó ese valor porque en algunos de los casos no se llegaba a un número mayor que ese, lo cual ocasionaba

que el contador no se reiniciara su valor y continuara incrementándose. Dicha instrucción se muestra en la figura 17 que se presenta a continuación.

```
if (residuo >= 980)
{
NF= 0;
}
delay(20);
```

Fuente creación propia

Figura 17. Reinicio del contador.

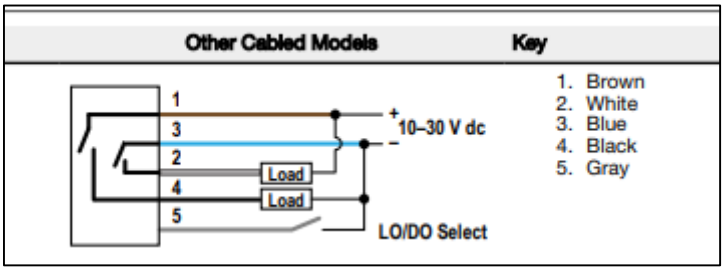
Además, como se muestra en la figura 17 se pone un delay de 20 milisegundos esto para dar un descanso al programa, siempre es recomendable un delay en los programas para su mejor funcionamiento, con lo hecho anteriormente se tiene un contador de flancos, así como la manera de reiniciarlo, ahora lo que prosigue es enviar dicho al puerto serial con la instrucción de la figura 18.

```
Serial.print(NF);
```

Fuente creación propia

Figura 18. Envió de dato al puerto serial.

Para detectar los flancos se hace uso del sensor de la marca Banner QS30 el cual se conecta de la siguiente manera que se muestra en la figura 18.

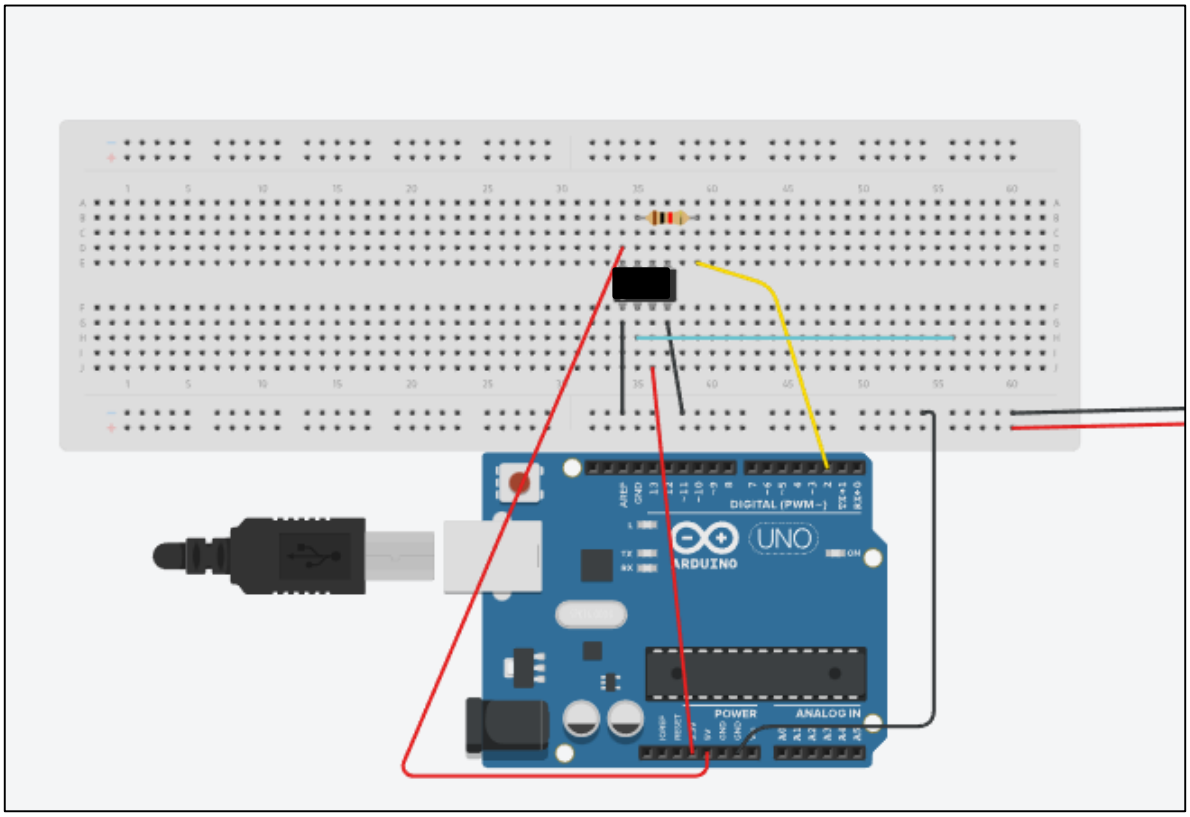


Fuente Banner QS30 data sheet.

Figura 19. Conexión del sensor Banner.

Como se muestra en la figura anterior se alimenta con un voltaje de 10 a 30 volts por los pines 1 y 3, este señor tiene dos maneras de operar una es como detector cuando se produce la reflexión se produce el flanco y la otra que es siempre

detectando reflexión y en cuanto esta se interrumpe manda el flanco, en el caso de este proyecto el interés es de detectar los flancos cuando se produce una reflexión, por eso se usa la salida 4 de color negro, pero existe el inconveniente de que la placa Arduino sus entradas solo van del 0 al 5 y es por ello que es que es necesario bajar el voltaje de los flancos, es aquí donde se hace uso del comparador de voltaje LM311P, cuya conexión se muestra en la figura 20 de a continuación.



Fuente creación propia

Figura 20. Circuito para LM311p.

En el circuito anterior las conexiones son: pin uno a tierra, pin dos (cable azul) es la entrada proveniente del sensor, pin tres es la entrada de 3.3 volts proveniente del Arduino, pin cuatro también se conecta a tierra, pin siete con una resistencia a la salida (cable amarillo) es la salida del comparador al pin 2 del Arduino y finalmente se alimenta con 5 volts por el pin ocho proveniente del Arduino. En este caso el funcionamiento es que si el voltaje es mayor a los 3.3 que se colocan como

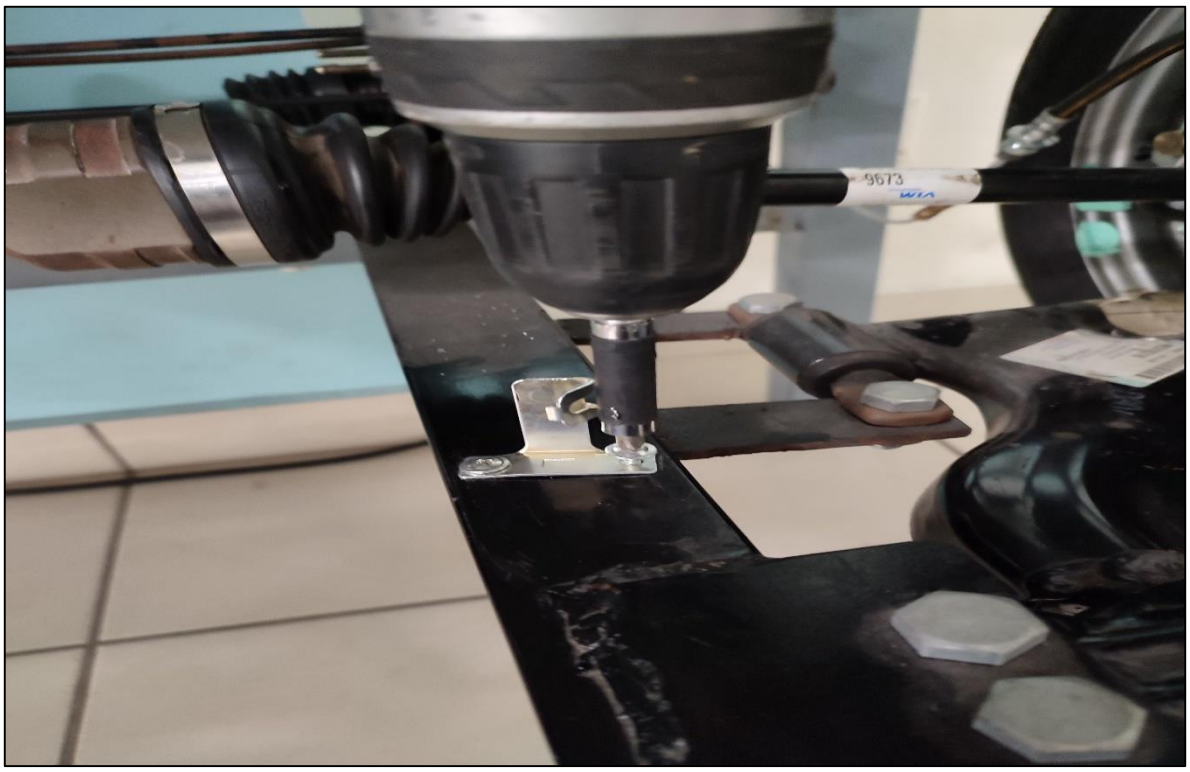
referencia en el pin tres el comparador a la salida nos dará 5 volts, que es un valor con el cual, si puede trabajar Arduino, de esta manera la placa Arduino ya puede trabajar con los flancos del sensor reflectivo.

Con lo anterior realizado lo siguiente es colocar los reflectores en el rin de la llanta, los cuales fueron colocados a una distancia de aproximadamente 90° o $\pi/2$ radianes, la figura 21 de a continuación muestra la colocación de los reflectores en la circunferencia del rin.

Fuente creación propia

Figura 21. Reflector en la llanta.

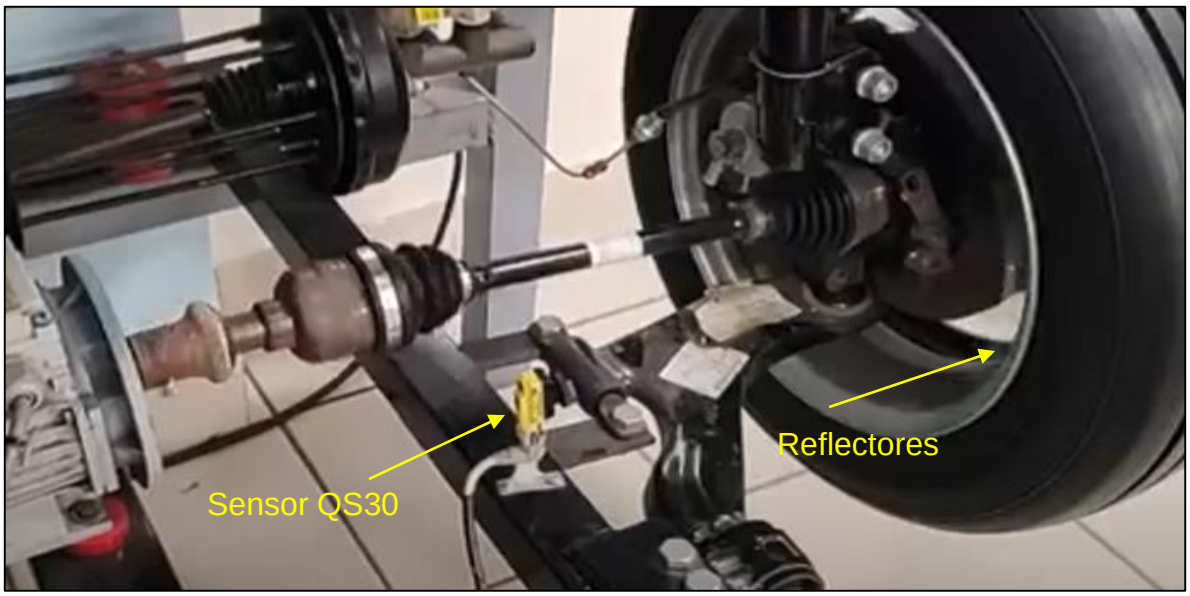
Para evitar que se pueda mover el sensor por las vibraciones mismas de la planta y esto lleve a que el láser deje de detectar los reflectores, se colocó una base adecuada para el sensor, misma que fue fijada a la estructura de la planta por medio de dos tortillos (ver figura 22).



Fuente creación propia

Figura 22. Base para el sensor reflectivo Banner.

El funcionamiento dentro de la planta es como se muestra a continuación en la figura 23 donde se aprecia el sensor detectando los reflectores a gran velocidad.



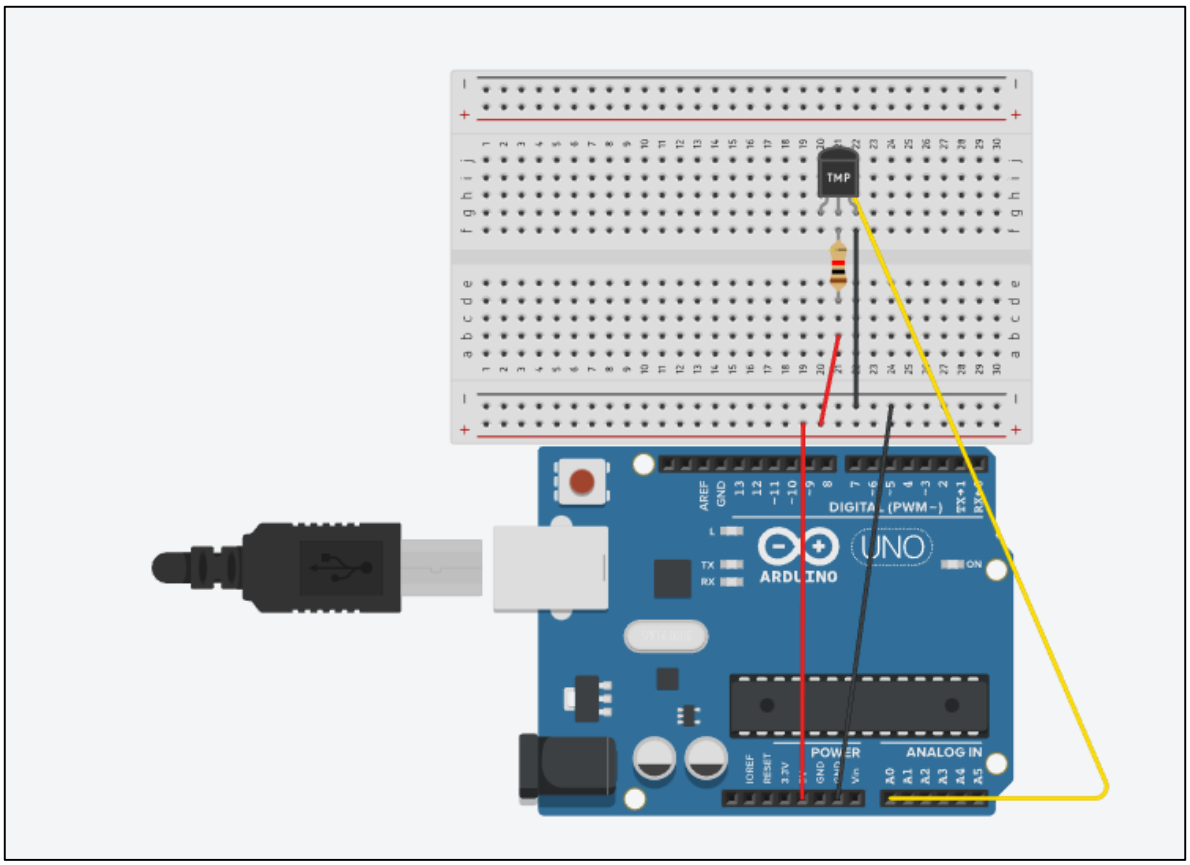
Fuente creación propia

Figura 23. Sensor y reflectores.

Con todo lo anterior se logra la detección de los flancos para poder calcular la velocidad de giro de la rueda.

Obtención de temperatura

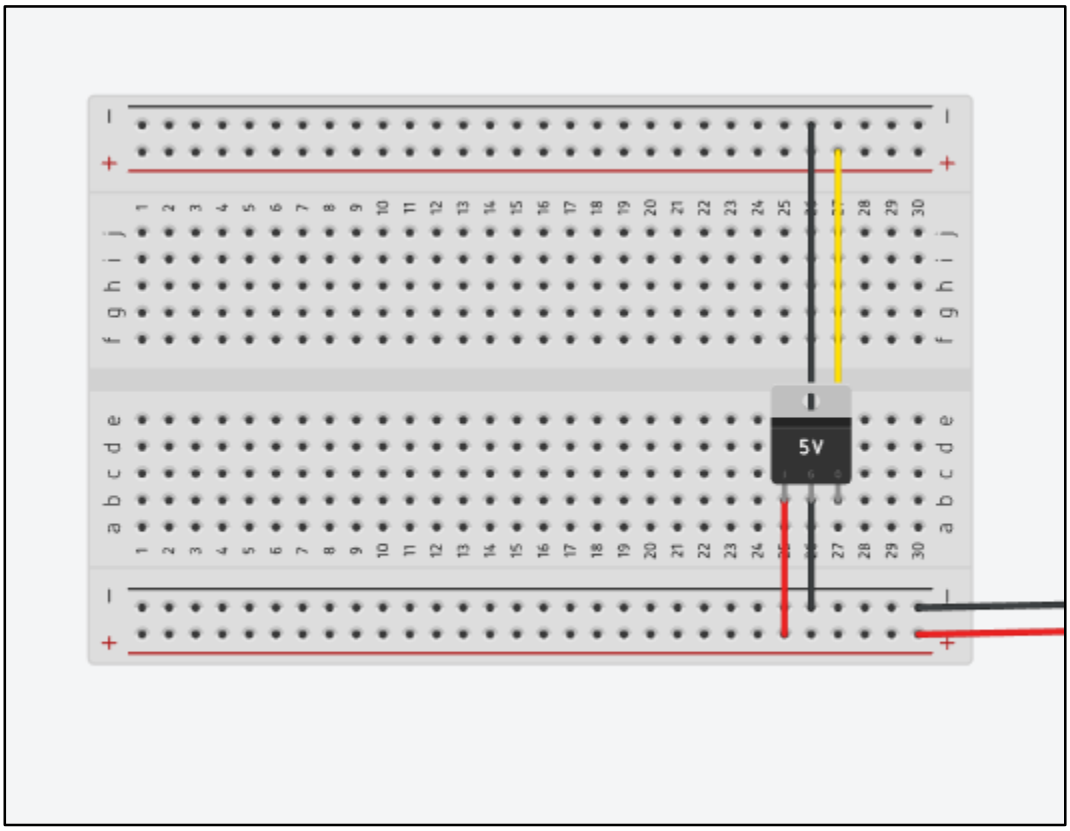
En el caso de la temperatura de la balata esta se obtiene gracias a un sensor de temperatura LM335, el cual se conecta de la manera que se muestra en la figura 23 a continuación.



Fuente creación propia

Figura 24. Circuito para sensor de temperatura.

En donde las conexiones son usando el A0 para registrar el dato medido, y la alimentación del sensor es por medio de la salida de 5 volts del Arduino y de GND del mismo Arduino, pero buscando no demandar tanto al Arduino se colocó un regulador de voltaje para bajar el voltaje de 12 volts que es con lo que trabaja el sensor reflectivo a 5 volts para alimentar el sensor de temperatura e incluso al Arduino mismo, para ellos se usó un regulador de voltaje LM7805 con un circuito como el que se muestra en la figura 25.



Fuente creación propia

Figura 25. Circuito regulador de voltaje.

La conexión es simple, en la parte inferior del protoboard se tienen 12 volts y de ahí se alimenta el regulador para tener a la salida 5 volts, los cuales con cableados a la parte superior del protoboard para poder ser usados para alimentar el sensor de temperatura y al comparador e incluso para alimentar la placa Arduino.

Con el circuito alambrado, lo siguiente es la programación en la interfase de desarrollo (IDE) de Arduino para poder leer el valor medido, dentro de la IDE se crea una variable al inicio del programa de tipo *double* con el nombre de Temp, posterior a la declaración solo basta la asignación de valor a dicha variable con la instrucción de analogRead dentro del Void loop, quedando de la manera que ilustra la figura 26.

```
Temp = analogRead(A0);
```

Fuente creación propia

Figura 26. Lectura del valor de temperatura.

Y por último se manda imprimir un punto y coma para separar los valores de número de flancos del valor de la temperatura, mismos que se manda imprimir después del punto y coma por el puerto serial de la siguiente manera que se observa en la figura 27.

```
Serial.print(NF);
Serial.print(";");
Serial.println(Temp);
```

Fuente creación propia

Figura 27.Valores a imprimir por puerto serial.

Comunicación serial

Dentro de este proyecto la visualización de los parámetros medidos era algo de suma importancia, fue por ello que se tuvo que buscar opciones para ver cuál era la más conveniente, y entre esas opciones estaba LabVIEW, que es un programa de programación visual.

Una vez que se eligió a LabVIEW como el lugar donde se desplegarían los parámetros el reto era lograr comunicación entre el software y la placa Arduino uno, fue entonces que buscando información al respecto se encontró algo similar a lo que se pretendía hacer en este proyecto y tomando como ejemplo a (Arduinolover, 2017) se adaptó lo hecho por el para las necesidades de este proyecto.

Como en el trabajo antes mencionado, lo primero es buscar dentro de nuestro software de LabVIEW si cuenta con los complementos necesarios para esta tarea

los cuales son: NI visa y VI package manager, en caso de no contar con los mismos se pueden obtener de la página de National instruments. Con la comprobación de los complementos ahora sí se procede a realizar la comunicación entre la placa Arduino uno y LabVIEW.

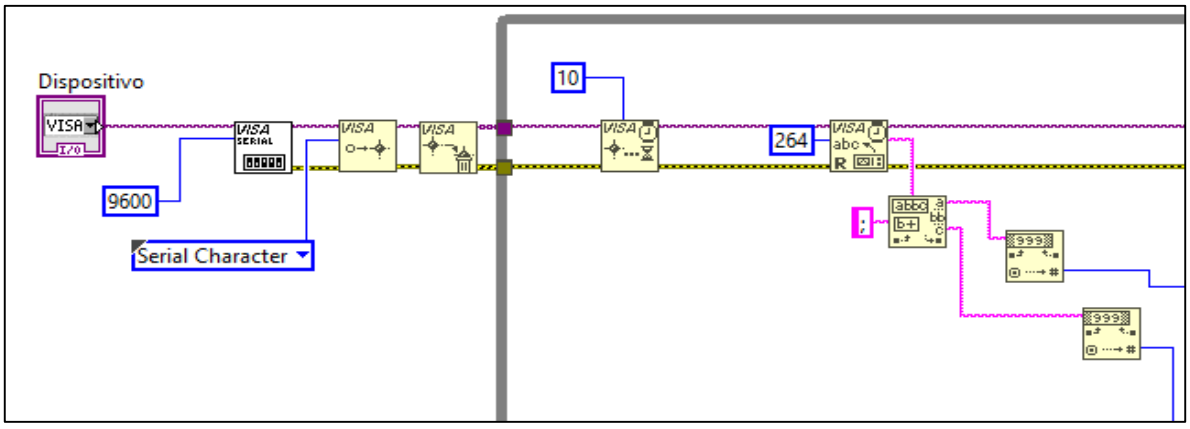
Lo primero es considerar la velocidad de comunicación misma que ya fue declarada en la primera sección de este capítulo, hecho eso se procede a declarar dentro de la plataforma Arduino cuáles son los datos que se quieren mandar hacia el puerto serial, mismos datos que normalmente son desplegados en el monitor serial, pero en este caso serán leídos en LabVIEW, se mandaron a imprimir por el puerto serial los datos de velocidad, se separa un dato del siguiente con un punto y coma, por último se manda el valor de la temperatura, en este caso en particular el dato de la temperatura se manda con la instrucción de Serial.println para que además se cree un salto de línea y termine esa cadena de caracteres, es la diferencia ya que en el caso del valor de velocidad y del punto y coma la instrucción solo fue Serial.print, esa es la diferencia únicamente .

Hecho lo anterior en Arduino se pasa ahora a LabVIEW, en el cual se crea un nuevo VI, después se empieza a programar en el diagrama de bloques utilizando las herramientas de serial y de visa dentro de la carpeta de instrument I/0.

Lo primero a colocar es el cuadro para configurar el puerto serial, que será donde se colocará la velocidad y el puerto a utilizar, para los cuales se le crea un control para el puerto serial y una constante para la velocidad. Después de esto en la carpeta de visa en avanzado se busca la opción de enable event, esta instrucción se conecta con el anterior cableando y uniendo las entradas y salidas de error y de visa resourse name, también se crea un control para indicar que tipo de evento queremos utilizar en este caso será serial Character. Para evitar que otro evento que no sea el seleccionado genere ruido o interferencias se coloca un Visa discard events, ahora se crea un lazo while que será donde sucederá el resto del programa, dentro del lazo while se coloca la instrucción visa wait on event al cual se le puede crear una constante para el tiempo que espere o el tamaño del dato a recibir y la

instrucción de visa read, mismas que se cablean a las anteriores, y para finalizar esta parte se cierra la comunicación con la instrucción de visa close.

Ahora ya se tiene la forma de obtener los datos a recibir, pero vienen todos juntos dentro de una cadena de caracteres es por ello que es necesario separarlos y aquí es donde es de gran ayuda el punto y que se mandó entre los dos datos, para separarlos se utiliza la instrucción de Match pattern para buscar el punto y coma y obtener el dato que esta antes y el que esta después del punto y coma para separar el dato de velocidad y temperatura, pero cabe recalcar que la comunicación entre Arduino y LabVIEW se realiza por medio de caracteres es por ello que es necesario colocar un convertidor de string a número para poder utilizar y visualizar los datos leídos, esto se realiza con un convertidor para cada uno de los datos uno para número de flancos y uno para temperatura, y como primero se envía el número de flancos este es el dato que esta antes del punto y coma y se ahí se cablea al convertidor y de igual manera de la salida de después del punto y coma que se busca en la cadena de caracteres se conecta al convertidor para obtener el dato de temperatura. Haciendo lo anterior ya se tienen listos los datos para su procesamiento dentro del mismo VI de LabVIEW para obtener los datos de interés como lo son la velocidad y la temperatura. Quedando todo lo anterior de la manera que ilustra la figura 20 de a continuación.



Fuente creación propia.

Figura 28. Configuración de comunicación serial.

Con lo anterior realizado ya es posible la comunicación entre Arduino y LabVIEW esta es una de las maneras en que puede realizarse, aunque existen otras opciones, con esto ya LabVIEW es capaz de leer datos desde Arduino ahora lo que sigue es escribir datos desde LabVIEW a Arduino, esto se realizara más adelante dentro de este mismo capítulo. Lo que sigue es procesar los datos obtenidos para obtener las variables de interés que en este caso son velocidad y temperatura.

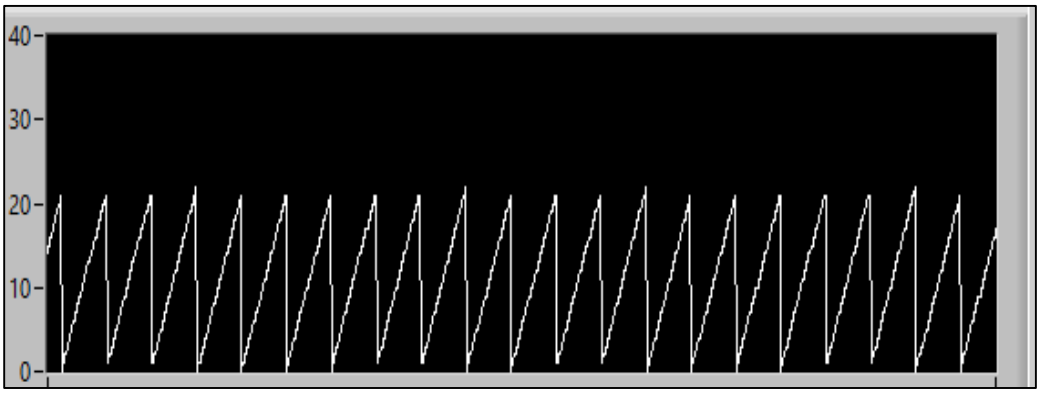
Obtención de velocidad

En esta parte del proyecto, se marca una diferencia importante con los proyectos previos. Ahora el sensor es real y no emulado por un generador de funciones, debido a esto el programa para la adquisición de la velocidad incluye rasgos completamente nuevos.

En el proyecto previo, el medidor de velocidad de la rueda también utilizaba interrupciones en la placa Arduino, pero en este caso se usaban para medir el tiempo que transcurría entre cada flanco que las producía, con el inverso de este delta se sacaba la frecuencia y luego el valor de la velocidad angular, de ahí la velocidad lineal multiplicando la velocidad angular por el radio de la llanta. Este medidor funcionó bien cuando los pulsos que emulaban la rueda girando era una señal dada por generador de funciones, pero una vez puesto en operación sobre la planta con el sensor real se encontró que tenía una falla, pues cuando la rueda se detenía la placa se quedaba con el último dato leído porque ya no llegaban flancos y la velocidad medida nunca llegaba a cero. A partir de este hecho, se decidió realizar un nuevo programa que no se basara en los tiempos entre flancos porque en la práctica los flancos desaparecerían al detener el giro.

Una parte de la intención de adquirir la velocidad con el Arduino UNO es para minimizar costos, sorteando sus limitaciones tanto para adquisición de señales como para su procesamiento, razón por la cual en este proyecto se plantea usarlo solo para contar el número de flancos y enseguida transmitirlo a una PC con LabVIEW. La velocidad de la rueda se calcula en dicha PC a partir de la señal que

se forma con los datos del contador programado en el Arduino, es decir, la velocidad se estimará por el número de flancos en un intervalo de tiempo fijo. Al monitorear el número de flancos leídos por el puerto serial siempre se observará una gráfica tipo diente de sierra, esto porque la placa se programa para contar los flancos en forma ascendente en intervalos de 1 segundo, al cumplirse este tiempo el contador se reinicia y el ciclo se repite (ver programa en el Anexo A). La figura 29 muestra una gráfica que implica una velocidad aproximadamente constante pues el valor máximo se mantiene alrededor de los 24 flancos en cada ciclo.



Fuente creación propia

Figura 29. Gráfica de número de flancos en función del tiempo, con periodo T=1seg.

La velocidad de la rueda está asociada con el máximo de flancos/seg de la señal por lo tanto en LabVIEW se tiene que obtener este valor y luego convertirlo a km/hr. En este punto, cabe hacer notar que el cálculo de velocidad se hace en la PC y no en el Arduino, lo cual evita el uso de recursos en la placa y le permite atender el conteo de manera prioritaria.

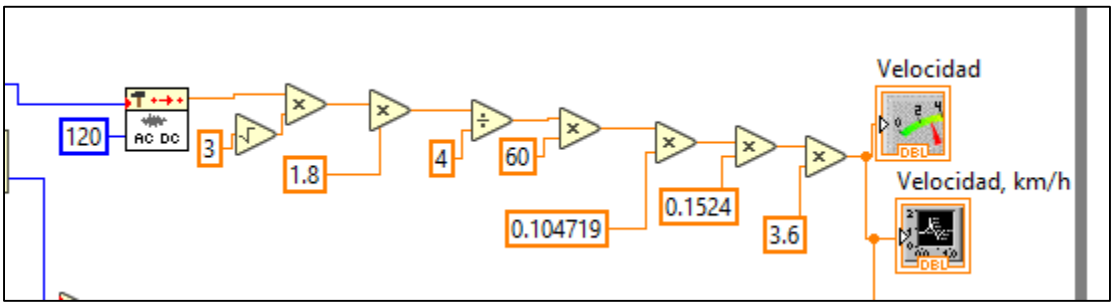
LabVIEW cuenta con un estimador de valores AC y DC para señales (AC & DC Estimator PtByPt.vi), su funcionamiento es tal que calcula el valor RMS y el valor de DC a través de N muestras de la señal de entrada. En nuestro caso se determinó experimentalmente que N=120 siempre produce una estimación aceptable del valor RMS para el rango de velocidades que se usa en la planta.

En general el máximo en la señal lo podemos escribir como la suma del valor pico más el valor de CD, para una onda de diente de sierra el valor pico está dado por $Vp = \sqrt{3} * V_{RMS}$. Por lo tanto, el máximo de nuestra señal debe ser $Vp + V_{CD}$. Esto funciona bien para el caso de la figura 29, en estas condiciones la primera versión del instrumento entrega como máximo resultante el valor de 23.5 flancos, lo cual concuerda con el máximo observado en promedio en la gráfica, este instrumento y calcula este valor en aproximadamente 2 segundos.

Al probar el instrumento con diversas velocidades de la rueda, cuando la velocidad aumenta el tiempo de cálculo aumenta y el resultado del máximo se aleja de la realidad, esto requiere que la consideración de CD no se sume, en su lugar se determinó experimentalmente que funciona mejor multiplicar un factor de 1.8 al Vp, con este valor de ajuste el máximo calculado siempre coincide dentro de un 10% con el valor gráfico para todo el rango de velocidades manejados en la planta.

Una vez obtenido el número de flancos por segundo que equivale al máximo calculado hay que determinar la velocidad de la rueda en Km/hr, estos son cálculos simples que se describen a continuación. Una revolución de la llanta son cuatro flancos en la señal porque son cuatro reflectores en el perímetro del rin, por eso el máximo calculado se divide entre 4, obteniéndose revoluciones/seg, de ahí se multiplica por 60 para tener RPM, enseguida se multiplica por $\pi/30 = 0.104719$ para obtener la velocidad angular en rad/seg, con la velocidad angular se obtiene la velocidad en metros por segundo al multiplicar la velocidad angular por el radio de la rueda, dicho valor en el caso de la planta es de 0.1524 m, y para conocer la velocidad en kilómetros por hora basta con multiplicar la velocidad lineal en m/s por 3.6.

Para la mejor visualización de los resultados se colocó un indicador de la velocidad en Km/hr y una gráfica para observar de manera visual los cambios que se presentan en función del tiempo, todo lo anterior dentro del VI se observa como lo ilustra la figura 30.

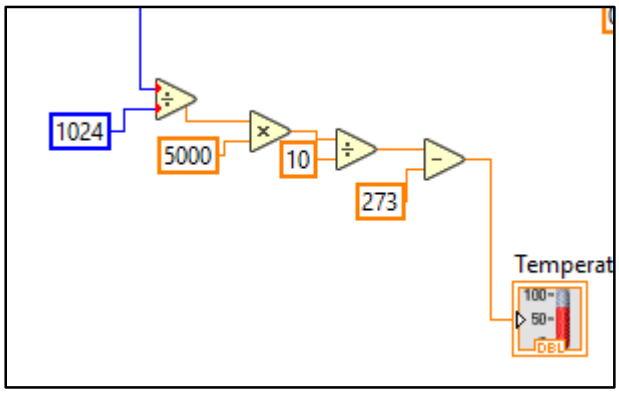


Fuente creación propia

Figura 30. Procesamiento de número de flancos a km/h

Procesamiento de la temperatura

En el caso de la temperatura se cuenta con el dato leído por el sensor y enviado por el puerto serial al VI de LabVIEW, solo basta procesar dicho dato para convertirlo en el dato de nuestro interés, el cual es en grados Celsius, para ello lo primero es dividir el dato entre 1024 que es el nivel máximo que se puede tener, después se multiplica por 5000 y como este sensor trabaja en grados Fahrenheit para obtenerlos de divide entre 10, pero como para nuestro caso queremos grados Celsius a este dato se le resta 273 y con esto ya se tiene el valor de temperatura en grados Celsius, los cuales son mostrados en un termómetro en el panel frontal del VI, dicha programación se ilustra en la figura 31.



Fuente creación propia

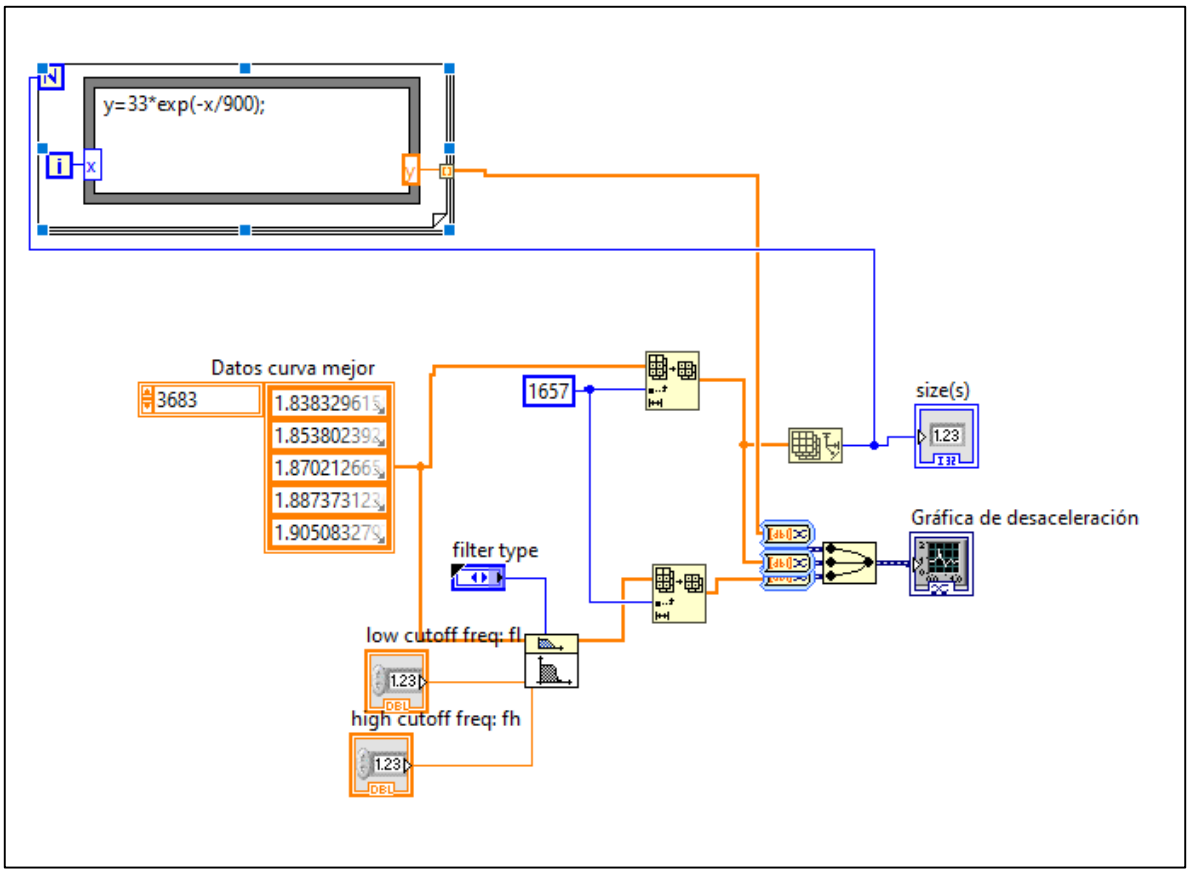
Figura 31. Procesamiento del dato para temperatura.

Obtención de grafica de desaceleración natural

Para obtener los datos para ver la desaceleración natural de la planta se almacenaron los datos en un arreglo para su procesamiento en otro VI aparte para conocer el comportamiento del sistema de manera natural, una vez copiados dichos datos en un nuevo VI por medio de un filtro pasa bajas para evitar los rizos que suele presentar la línea de comportamiento del sistema, después del filtro se coloca un Array subset para solamente obtener la curva en el momento en el cual empieza a desacelerar la planta y no tener todos los datos desde que estaba estable el sistema, después de esto se utiliza un Merge signals para poder graficar tres señales al mismo tiempo, después de esto se coloca una waveform para apreciar las gráficas en el panel frontal.

Para poder comparar como es el comportamiento de la señal filtrada en comparación a la normal se grafica de igual manera a partir también del dato en cuanto empezó a descender la velocidad, y como ya se tiene la opción de graficar varias señales al mismo tiempo es cuestión solo de conectar esta otra señal al Merge signals.

Como algo adicional se intentó buscar un modelo matemático que se asemejara al comportamiento de la planta de frenado para ello se hizo uso de una estructura For y de un nodo de fórmula, en el cual el valor de x será el número de la interacción que realice el ciclo For, para el número de veces a realizar el ciclo este está dado por el tamaño del arreglo de datos con el cual se está trabajando, por último la salida o valor de Y será el dato que se conecte al Merge signals para al mismo tiempo graficar tanto la señal real, la filtrada y al del modelo matemático y apreciar así el comportamiento de la planta al momento de desacelerar de manera natural sin la intervención se ninguna fuerza externa, dicha programación dentro del diagrama de bloques de ilustra en la figura 32.



Fuente creación propia

Figura 32. Obtención de graficas de desaceleración.

Con lo anterior se puede hacer una comparación entre las diversas señales, las cuales ayudan al mejor entendimiento del sistema.

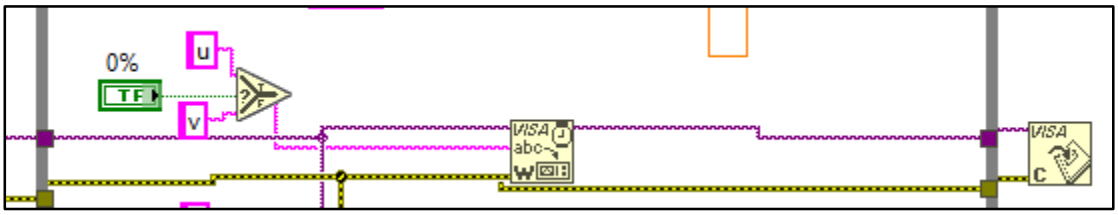
Programación de la válvula

En el caso de la válvula se decidió operarse a través de otro Arduino y otro VI para no saturar o demandar de más a una sola placa Arduino y llegue a fallar o detenerse en un instante dado, en el caso de la programación de la válvula lo primero al igual que en el VI para velocidad y temperatura lo primero es configurar el puerto por el cual será la comunicación, lo que si cambia es la instrucción porque ahora no se leerá sino que se va escribir desde LabVIEW hasta la placa Arduino un valor

determinado para poder escribir un determinado valor de modulación PWM para la activación de la válvula en determinadas circunstancias.

Para el caso de la activación de la válvula se empezó primero con la programación en LabVIEW para de ahí pasar a Arduino. En este caso en vez de leer datos por el puerto serial se escriben datos a través de para llegar a Arduino y realizar las acciones correspondientes para escribir el valor de PWM de acuerdo a los que se le indique y así se active la válvula desde un 0 hasta el 100%.

Lo primero es como en la sección anterior conectar al dispositivo y el error de salida del dispositivo hacia el ciclo while, ya que se estará trabajando por el mismo puerto y la misma velocidad de la sección anterior, posteriormente a esto en el diagrama de bloques se coloca un interruptor de dos posiciones (verdadero y falso), este a su vez va conectado hacia un select, esto con la finalidad de crear constantes de caracteres para que mande en el caso que sea verdadero o falso para esto hacemos uso de las letras del alfabeto, esto se realiza porque recordemos que la comunicación entre Arduino y LabVIEW es por caracteres, por ello el uso de las letras en lugar de escribir un 1 o un 0, en este caso la salida del select va unida a una instrucción que también se encuentra en las herramientas de visa y se conoce como visa write, porque ahora se vas escribir un dato o varios, dicha instrucción se conecta al dispositivo fuente y al error del mismo, hecho esto lo único que falta de igual manera es cerrar la comunicación con un close, en el caso para la válvula se repite esta acción 11 veces ya que se parte desde el 0% y llega al 100%, la programación en todos los casos es la misma solo cambian las constantes alfabéticas, las cuales empiezan desde la a para poder diferenciar cada uno de los casos de la modulación PWM que queremos ejercer en la válvula. En la figura 33 de a continuación se ilustra de manera un poco más visual para la mejor compresión lo descrito en el presente párrafo.



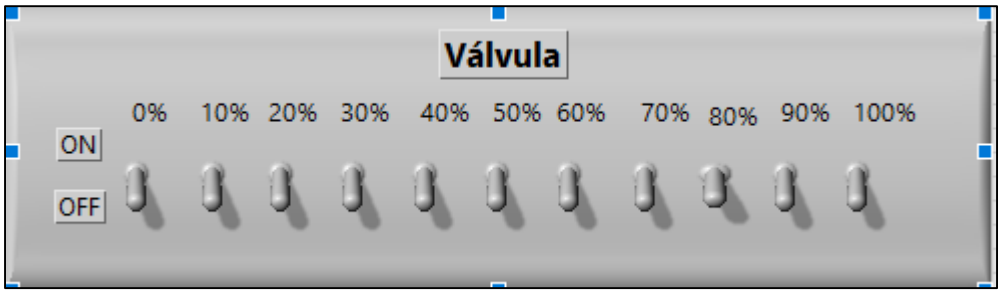
Fuente creación propia

Figura 33. Programación para escribir un dato.

Por último, se debe crear un botón de stop para el programa, ya que esta es la manera correcta de pararlo, ya que si no se detiene de esta manera se va encontrar el puerto serial ocupado y no va poder ser utilizado para otras funciones como serian subir a la placa Arduino uno modificaciones hechas al programa, de ahí la importancia de cerrar y detener la comunicación serial con ese botón, cuya creación y conexión es simple, basta con crear el botón en el diagrama de bloques y conectarlo a la condición de paro del programa.

En este caso se hizo una adaptación de lo hecho por (Romero, 2019), el cual hace uso de un VI para controlar un servomotor que es una de las aplicaciones más comunes para una modulación PWM, pero en este proyecto se usó para una válvula de ahí las modificaciones, ya que no fue la misma aplicación.

Dentro de LabVIEW los controles del VI quedaron de la amanera que ilustra a continuación la figura 34.



Fuente creación propia

Figura 34. Controles para la válvula.

Hecho lo anterior en LabVIEW se procede a pasar ahora a la plataforma de Arduino para continuar la programación, ya que este será el encargado de escribir el valor

de PWM que se le indiqué por medio del VI y la comunicación serial entre ambos, ya en la plataforma de Arduino, lo primero a realizar es crear una variable para almacenar el valor que se va leer de LabVIEW, para este caso se creó la variable de valor almacenar ahí el dato que se enviara.

Hecho lo anterior haciendo uso de una condicional IF dentro del void loop, se da la condición que si la comunicación serial esté disponible la variable valor tomara el valor del dato leído mediante la instrucción de Serial.read. a partir de esto haciendo uso de la misma estructura IF se crea una condición para cada uno de los valores posibles que se pueden obtener a partir del VI, por ejemplo, en el primer caso para una señal del 10%, se creó la condición de que si la variable valor tenía como dato una “a” mediante la función de analogwrite y por un pin especifico, se escriba un valor de 26 que es el equivalente al 10% de la señal PWM, cuyo valor máximo es de 255, y como toda instrucción o línea en Arduino hay que finalizar con punto y coma. Lo anterior dentro de la plataforma de Arduino quedaría de la siguiente manera que muestra la figura 35.

```
void loop() {

if (Serial.available())
{
  valor = Serial.read();

  if (valor=='a')
  {
    analogWrite(11,26);      //10%
    }
```

Fuente creación propia

Figura 35. Programación para modulación PWM del 10%.

Lo anterior se repite para cada uno de los casos solo cambiando la condición y el valor a escribir para cada caso, fue necesario escribir un valor de 0%, ya que si se colocaba dentro de una condición de else en cualquiera de los IF haría que no se escribiera el valor deseado, fue por ello que también se creó un condicional para 0 en LabVIEW y por supuesto en Arduino. Y esto sería lo hecho en programación en

los dos softwares. En el capítulo siguiente se mostrarán los resultados obtenidos con dichos programas.

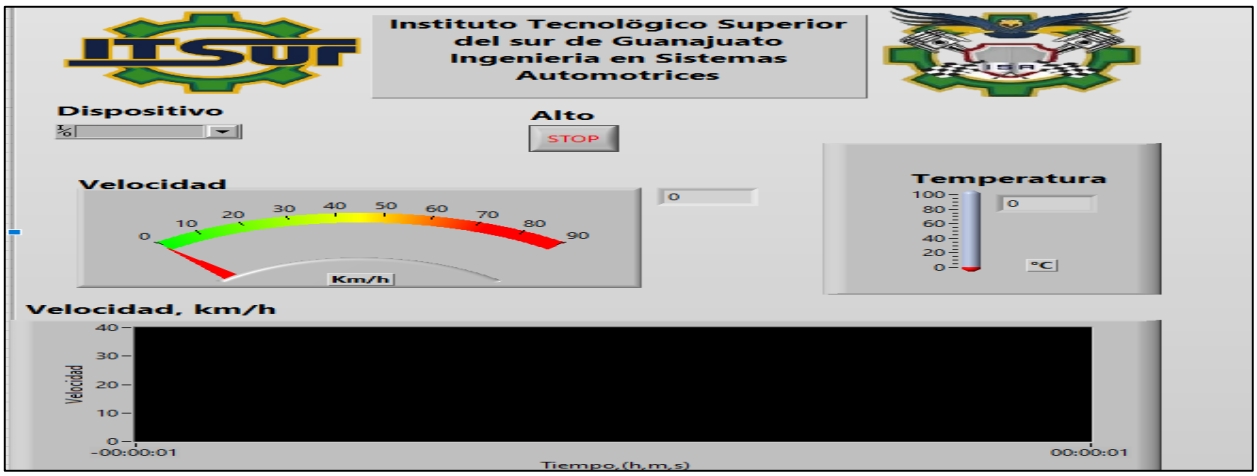
Capítulo 6

Resultados

Los resultados obtenidos dentro de la realización del presente proyecto se dividen en dos secciones, en la primera se muestran las mediciones realizadas con un tacómetro digital y su comparación con las mediciones obtenidas mediante el VI de LabVIEW y en la segunda sección se muestra el funcionamiento que tendrá la válvula mediante su operación controlada a través de otro VI de LabVIEW, mismo que opera de manera independiente al medidor de velocidad y temperatura.

Resultados de velocidad

Los resultados que se presentan a continuación cuenta con dos partes en la primera del lado izquierdo o inciso (a) se presenta la medida obtenida en la planta con un tacómetro digital, por su parte del lado derecho o inciso (b) se ilustran las mediciones realizadas con el VI creado para el proyecto, este VI además presenta un indicador para desplegar la velocidad de la planta, una gráfica para ilustrar el comportamiento de la misma, ya que suele presentar unos picos repentinos como se aprecia en figura 36 la el inciso (b) en su parte inferior en la gráfica de velocidad se aprecia un pico, y por último el VI presenta el valor de la temperatura.



Fuente creación propia

Figura 36. VI de velocidad y temperatura.

(a) **(b)**

Fuente creación propia

Figura 37. Medida de tacómetro(a) 182.6 RPM y medida VI (b) 181.42 RPM, con una velocidad de 10.82 km/h.

(a) **(b)**

Fuente creación propia

Figura 38. Medida de tacómetro (a) 30.7 RPM y medida VI (b) 30.92 RPM con una velocidad de 1.7 km/h

(a) **(b)**

Fuente creación propia

Figura 39. Medida tacómetro (a) 645.5 RPM y medida VI (b) 655.35 RPM con una velocidad de 37.81 km/h.

(a) **(b)**

Fuente creación propia

Figura 40. Medida tacómetro (a) 562.9 RPM y medida VI (b) 563.29 RPM con una velocidad de 32.27 km/h.

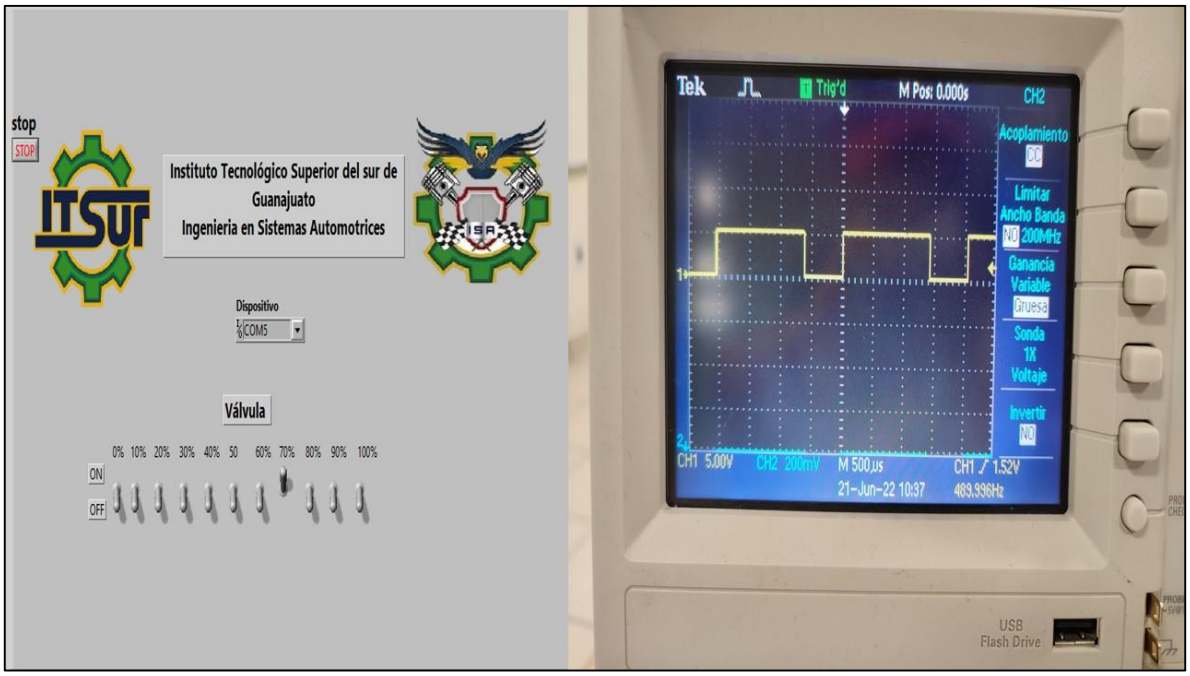
(a) **(b)**

Fuente creación propia

Figura 41. Medida de tacómetro (a) 314.9 RPM y medida del VI (b) 291.33 RPM a una velocidad de 16.34 km/h.

Resultados de la válvula

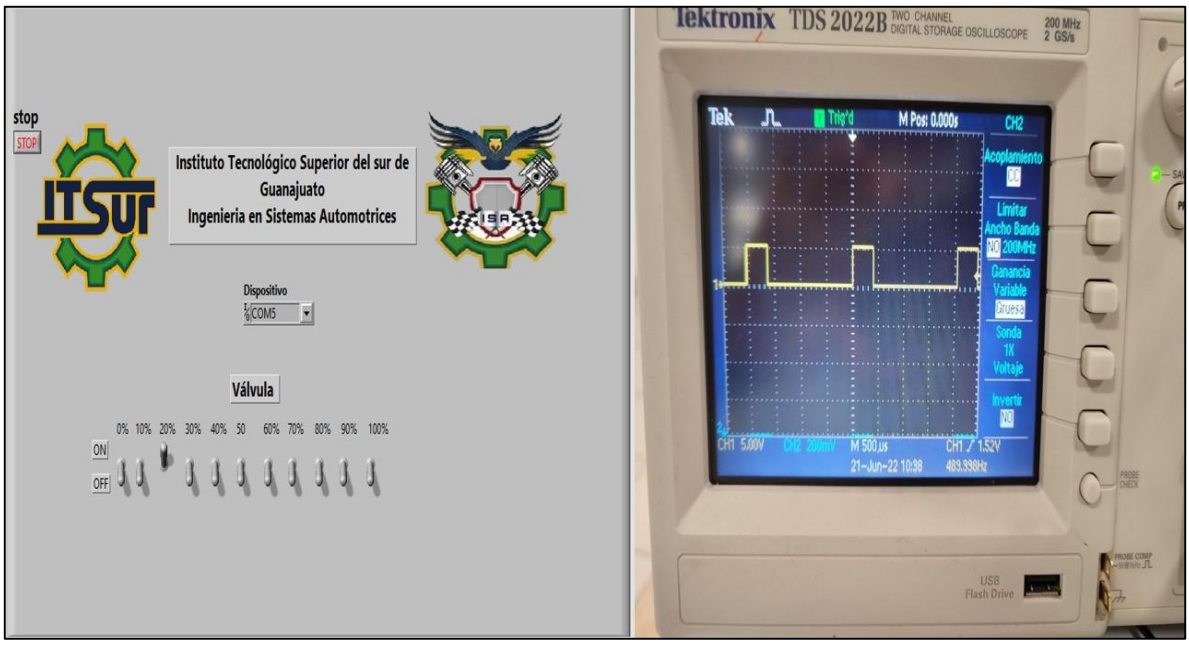
En esta segunda parte de los resultados corresponden a la válvula que se operara en la planta de igual manera a la sección anterior consta de dos incisos, el inciso (a) corresponde al lado izquierdo y es el botón del VI que activa a determinado porcentaje la válvula, por parte del lado derecho o inciso (b) muestra la señal de PWM que se tendrá a la salida, misma que es monitoreada a través de un osciloscopio, esto al no aún tener la válvula instalada dentro de la planta.



(a) **(b)**

Fuente creación propia

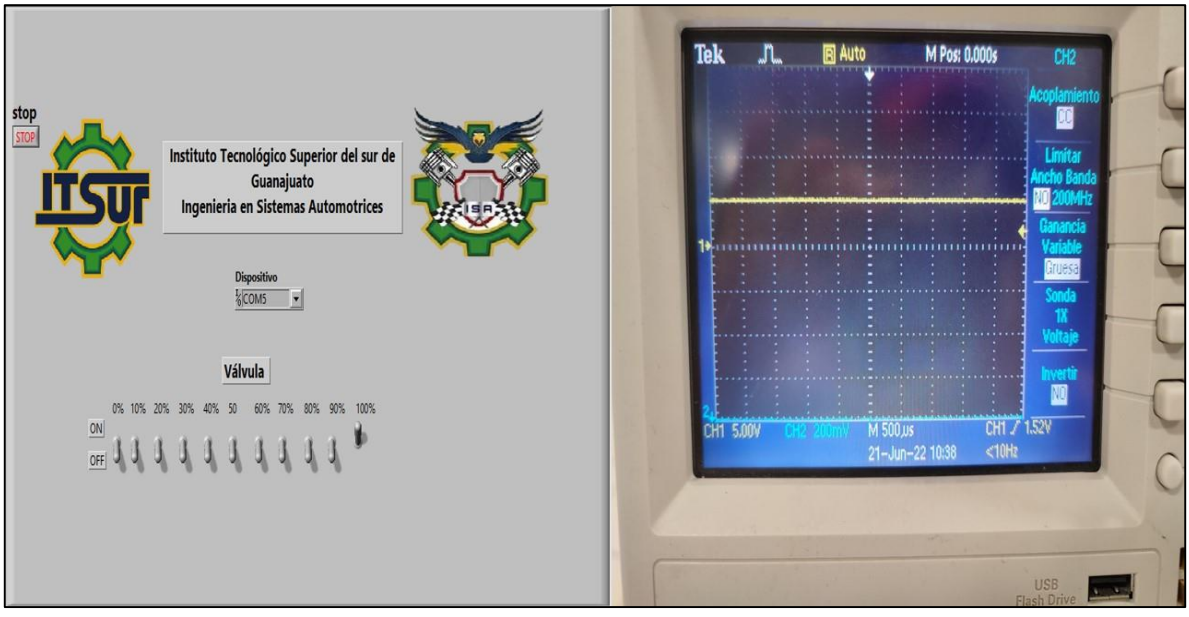
Figura 42. Inciso (a) VI en 70% salida PWM inciso (b).



(a) **(b)**

Fuente creación propia

Figura 43.Inciso (a) VI al 20 % y salida PWM inciso (b).



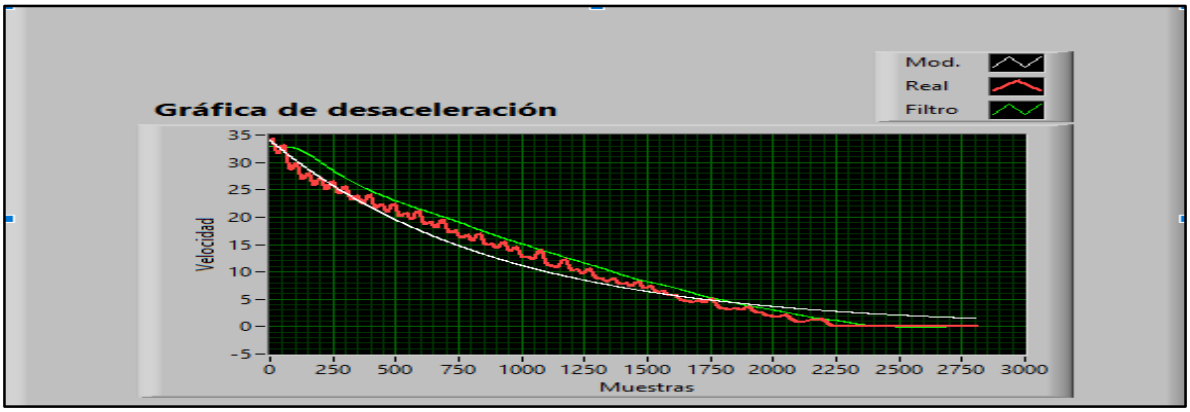
(a) **(b)**

Fuente creación propia

Figura 44. Inciso (a) VI al 100% y salida PWM inciso (b).

Graficas

Estas son las gráficas obtenidas de desaceleración natural, en color blanco modelo matemático, color rojo respuesta real y color verde grafica con filtro.



Fuente creación propia

Figura 45. Graficas obtenidas.

Capítulo 7

Análisis de Resultados

Para el mejor análisis e interpretación de los datos se colocaron dentro de la tabla de a continuación.

Tabla 3. Tabla comparativa de resultados.

Medición RPM en tacómetro	Medición RPM con VI	Diferencia entre mediciones	Velocidad medida en VI Km/h
30.7	30.97	0.87%	1.7
182.6	181.427	0.64%	10.82
314.9	291.33	8.09 %	16.34
562.9	563.29	0.06%	32.27
645.5	655.35	1.5%	37.81

Fuente creación propia

Como se muestra en la tabla anterior las mediciones entre ambos instrumentos no difieren mucho la una de la otra, la mayor diferencia que se encontró entre ambas mediciones fue del 8.09%, misma que aún se encuentra dentro del margen del 10% de error, en las demás mediciones la diferencia fue mínima en la mayoría de los casos no llego ni siquiera a la unidad, por lo cual las mediciones realizadas con el VI pueden considerarse como válidas y se comprueba la eficiencia del VI creado para este proyecto, en este caso el VI para velocidad y temperatura, el comparar ambas mediciones es de gran ayuda para demostrar la validez que tienen los datos que nos arroja el VI, es por ello que se puede decir que la velocidad máxima que se logra alcanzar en la planta de frenado es de 37.81 km/h, al llegar a dicha velocidad la planta de frenado comienza a generar vibración y a generar ruidos por lo que se denota el gran esfuerzo que realiza el motor para llegar a dicha velocidad.

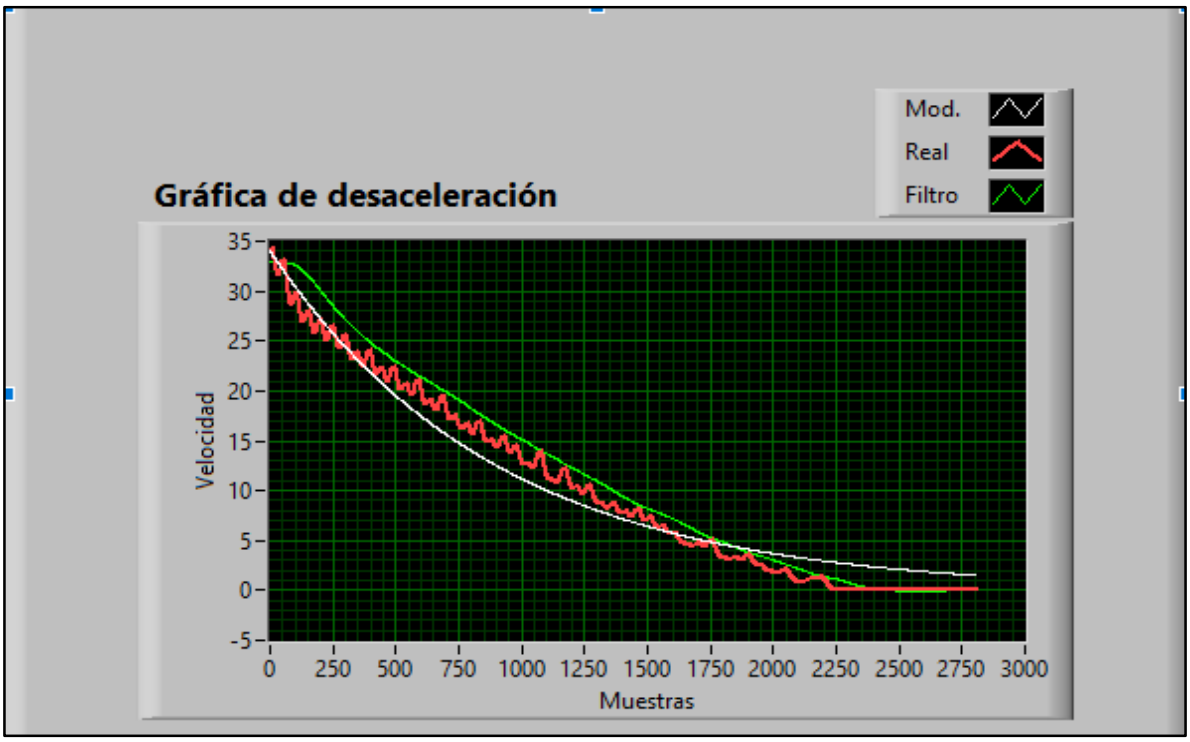
Se podría decir quizás que esta variación de medidas pueda deberse al delay que se coloca dentro del programa de Arduino para darle un descanso y no consumir de inmediato sus recursos, puede que en ese pequeño descanso se pierdan unos flancos, pero como se ve no es tan grande la diferencia de medidas, por lo tanto, este delay que si ayuda mucho al programa no genera cambios tan bruscos en las mediciones.

Como se aprecia con este nuevo programa ya no tenemos el problema de que la velocidad se quede con el último dato leído, ahora se tiene un programa capaz de llegar de una velocidad alta hasta cero, con esto se corrigió el problema que se tenía anteriormente, además con este nuevo programa se le exige menos a la tarjeta Arduino y se le da un pequeño delay para que no esté trabajando de manera continua.

Respecto a eso, este nuevo programa con esas nuevas incorporaciones del delay y el procesamiento de los datos en LabVIEW, se logró desaparecer otro problema que tenía el programa del proyecto anterior, el cual era que después de cierto tiempo el programa se congelaba y había que detener todo y empezar de nuevo, por ende no se podía tener la información completa. Ahora se tiene un programa que es capaz de trabajar de manera continua por un intervalo de tiempo amplio y que nos da información sobre una posible desaceleración completa, es por ello que este trabajo fue una gran mejora a lo hecho anteriormente.

Otro resultado importante obtenido fue la curva de desaceleración natural de la planta al llegar a su velocidad máxima que en este caso fue de 37.81 km/h, misma

que en base a los datos obtenidos se le aplico un filtro para mejorar su representación y obtener una línea más uniforme, además se buscó un modelado matemático que se asemejara lo más posible al comportamiento de la planta al momento de desacelerar de manera natural sin la aplicación de fuerza o de la activación del sistema mecánico de freno con balatas. A continuación la figura 46 ilustra el comportamiento de las gráficas.



Fuente creación propia

Figura 46. Graficas a analizar.

En la figura anterior se muestran tres graficas: la gráfica de la medición real (rojo), grafica con el filtro pasa bajas (verde) y el modelado matemático (blanco). Como se denota las tres son similares varían muy poco entre ellas, con esto se muestra como es el comportamiento de la planta al momento de desacelerar de manera natural, la cual gracias al modelo matemático que se asemeja a su comportamiento se denota que es una exponencial decreciente.

Con esta información y a partir de los cálculos realizados para encontrar el tiempo de muestreo se puede hacer una estimación del tiempo en el cual se detendrá la

planta de manera natural. En el eje x de la gráfica lo que se muestra es el número de muestras y tiempo, por lo tanto, con el dato de frecuencia de muestreo que se calculó en secciones anteriores y el número de muestras se podrá estimar el tiempo de frenado de manera natural de la planta que se realiza de la siguiente manera.

$$n\ de\ muestras = 2803\ muestras$$

Se hace uso de la frecuencia de muestreo calculada en secciones anteriores

$$frecuencia\ de\ muestreo = 0.0494\ muestras\ por\ segundo$$

$$tiempo\ de\ frenado = n\ de\ muestras * frecuencia\ de\ muestreo$$

(Ec.2)

$$tiempo\ de\ frenado = 2803\ muestras * 0.0494\ muestras * s$$

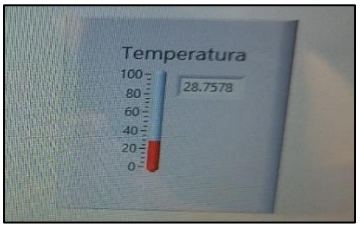
$$tiempo\ de\ frenado = 138.46\ s$$

$$tiempo\ de\ frenado = 2.30\ minutos$$

$$tiempo\ de\ frenado = 2\ minutos\ y\ 18\ segundos$$

Con los cálculos anteriores se puede estimar que de manera natural la planta de su velocidad máxima de 37.81 km/h para llegar a cero le tomara dos minutos y 18 segundos aproximadamente.

En cuanto a la temperatura los resultados no variaron ya que al no aplicarse ninguna fuerza a las balatas su temperatura no varió y se mantuvo constante de 28.75 °C todo el tiempo, y siempre fue la medición como se muestra en la figura 47 siguiente.

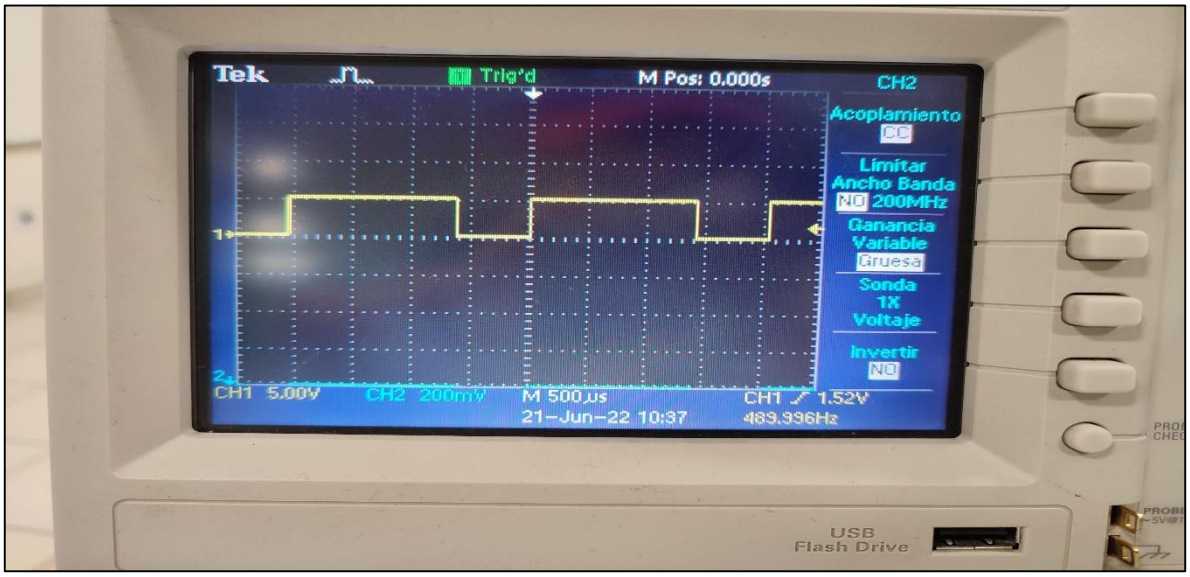


Fuente creación propia

Figura 47. Medida de temperatura.

En el caso de la válvula que es parte del segundo VI de los resultados obtenidos se puede decir lo siguiente.

En el primer caso que ilustra la figura de 48 a continuación, se obtiene una modulación PWM del 70%, esto quiere decir que la válvula permanecerá activa el 70% del tiempo y el otro 30% permanecerá apagada, con esto se esperaría que la velocidad en la planta disminuya de manera considerable, ya que la fuerza que el líquido de freno ejerza sobre las balatas será de magnitud considerable, ya que estará activa más de la mitad del tiempo, por lo que se puede pensar que la velocidad se reducirá y puede que llegue a 0 en un determinado intervalo de tiempo. Esta desaceleración podrá ser registrada en la gráfica del VI de velocidad, se apreciara de manera clara el momento en el cual se active la válvula y la velocidad comience a descender de manera gradual.

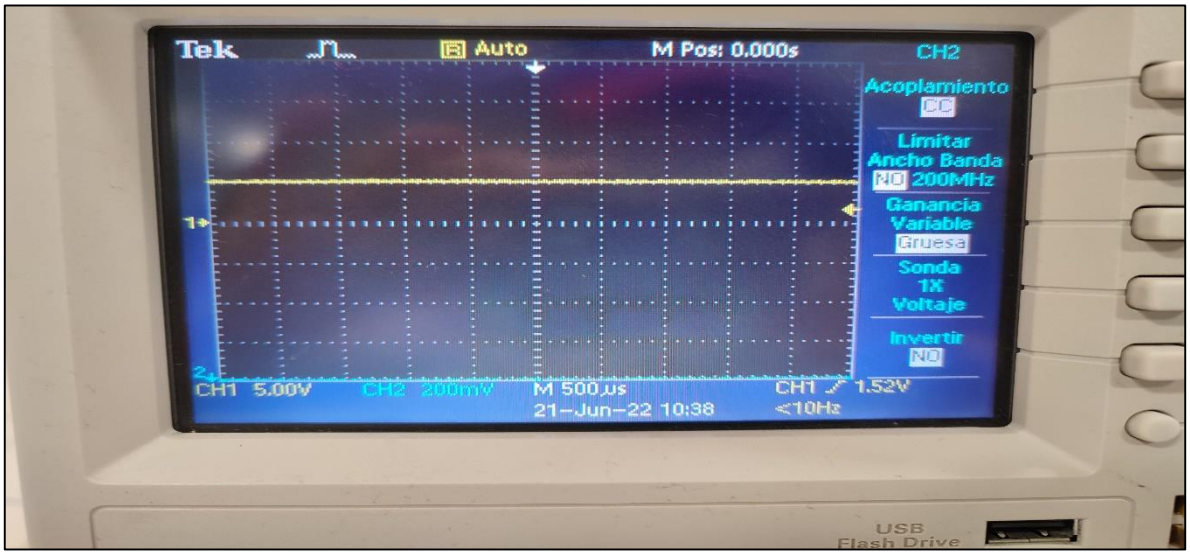


Fuente creación propia

Figura 48. Válvula al 70%.

En otro caso de los resultados que se tienen para la temperatura será una válvula activada el 100% del tiempo con lo cual no habrá tiempo de inactividad y la válvula

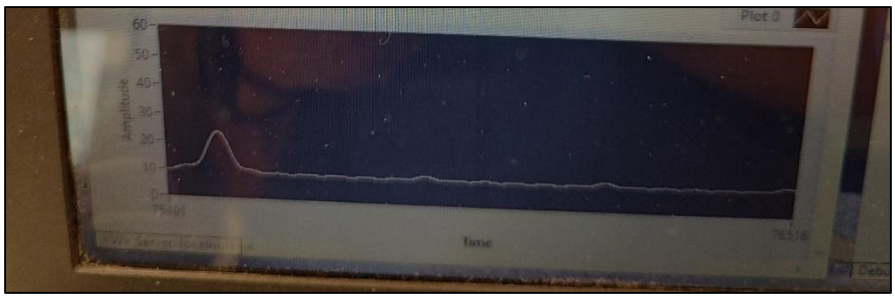
operara a toda su capacidad por lo que la velocidad descenderá de manera repentina y en unos instantes de tiempo llegara a cero, dicho intervalo de tiempo será menor que si se tuviera un porcentaje como el anterior del 70% u otro, dicha modulación de PWM se observa como lo muestra la siguiente figura 49 tomada de un osciloscopio.



Fuente creación propia

Figura 49. PWM al 100%.

Otro resultado que llamó la atención fueron picos como muestra la siguiente figura.



Fuente creación propia

Figura 50. Picos encontrados en la velocidad.

Estos picos se presentaban al aumentar la velocidad del motor por lo que deben deberse a él o a la parte mecánica de la planta por lo que es un área de oportunidad para mejorar y corregir en el futuro para el mejoramiento de la planta.

Capítulo 8

Conclusiones y trabajo a futuro

Con lo presentado en este documento se logra demostrar que es viable el uso de la placa Arduino UNO para el control y monitoreo de las condiciones de funcionamiento de la planta didáctica del sistema de frenado, la placa resulto ser eficiente al realizar las mediciones correspondientes y su transmisión vía puerto serial a un VI de LabVIEW para su procesamiento. Una de las ventajas más sobresalientes de esta placa Arduino UNO es su bajo costo el cual está entre los 200 y 300 pesos dependiendo del proveedor, con lo que es un micro controlador de fácil acceso para la mayoría de las personas, esto es una gran ventaja ya que no es necesario utilizar un PLC o una tarjeta de adquisición de datos de NI, ya que dichos equipos son de alto costo, son equipos de miles de pesos que pueden costar hasta doscientas veces más que un Arduino. Con este proyecto quedo demostrado su cabal funcionamiento al momento de escribir y leer datos por medio de un puerto serial, su comunicación con LabVIEW fue exitosa, no solo se pudo leer los datos que enviaba Arduino, sino también se pudo escribir una modulación PWM desde LabVIEW a la placa Arduino. Se puede utilizar un solo VI para leer y escribir, pero para no demandar tanto a una sola placa Arduino se recomienda una placa para adquirir y otra para emitir las señales hacia la planta.

Al comparar los resultados obtenidos con el instrumento de este proyecto con un tacómetro digital se demostró la eficacia del Arduino al detectar los flancos para de ahí calcular las RPM y la velocidad, ya que la variación en algunos casos fue mínima, y eso es de gran importancia, ya que el Arduino en costo también está por debajo del precio de un tacómetro y aun así logro resultados similares a pesar de no ser un dispositivo diseñado para dicha tarea como si lo es un tacómetro, gracias al uso de una placa como Arduino se puede obtener la información deseada sin una gran inversión de dinero, ya que esto abarata el costo de una planta como esta, ya

que en el mercado plantas similares a esta suelen tener un alto costo, su costo suele estar a partir de los dos mil euros. Con la placa Arduino se tiene un precio bajo por un alto grado de funcionalidad.

Al tener la adquisición con hardware de bajo costo sobre la planta didáctica, también se abre la posibilidad de que esto sirva para que los alumnos instalen su propia placa y pongan a prueba su propia programación, desarrollando mejores habilidades.

En cuanto al trabajo a futuro está la inquietud de probar con otro tipo de placa Arduino, tal como la placa Arduino Portenta, para saber si el aumento de costo (mayor a 150 dólares), asociado a sus mayores prestaciones implica mayor capacidad en la didáctica de la planta.

Otra área de oportunidad para trabajar son los picos que se encontraron al momento de incrementar la velocidad del motor, se puede buscar la causa de los mismos y tratar de eliminarlos o simplemente encontrar la razón por la cual se presentan, quizá el aspecto de la estructura de la planta sea la razón, se recomienda reducir la vibración que se presenta a altas velocidades pues puede causar el movimiento del sensor y que deje de detectar los reflectores, además puede ser peligroso para quienes estén cerca de la planta, en especial se recomienda revisar la mecánica en el eje que conecta el motor porque se oyen ruidos y se evidencia una alta fricción.

Por todo lo anterior se concluye que se logró cumplir los objetivos, se puede decir que el proyecto se logró con éxito. Está dicho que en la planta hay más por mejorar, pero todo va marchando bien y con lo realizado en este proyecto se tiene una planta didáctica de frenado más completa.

Referencias bibliográficas

Referencias

Arduino. (2022). *Arduino home*. Obtenido de Arduino home: https://arduino.cl/que-es-arduino/

Arduinolover. (octubre de 2017). *Youtube*. Obtenido de Youtube: https://www.youtube.com/watch?v=GQwSWOLFDkQ

Banner. (julio de 2022). Obtenido de Banner: https://www.bannerengineering.com/mx/es/products/sensors/photoelectric-sensors.html#all

De Lorenzo. (2022). *De Lorenzo*. Obtenido de De Lorenzo: https://delorenzoglobal.com/es/autotronica/frenado/

Digi-Key. (enero de 2022). *Digi-key*. Obtenido de Digi-key: https://www.digikey.com.mx/es/product-highlight/t/texas-instruments/lm335-temperature-sensor

Edibon. (marzo de 2022). *Edibon*. Obtenido de Edibon: https://www.edibon.com/es/equipo-de-freno-de-tambor

Geek factory. (2022). Obtenido de Geek factory: https://www.geekfactory.mx/tienda/componentes/reguladores-de-voltaje/lm7805-regulador-de-voltaje-5v-1a-to-220/

Guzmán, J. C. (2022). *Instrumentacion de planta didactica del sistema de frenado .* Uriangato: ITSUR.

Hidalgo, E. (marzo de 2022). *Diario de una ingeniera*. Obtenido de Diario de una ingeniera: https://himarele.wordpress.com/tag/labview-2/

Manuel Alejandro Pérez Martínez, Ó. S. (2019). *Pistas educativas*. Obtenido de Pistas educativas: http://itcelaya.edu.mx/ojs/index.php/pistas/article/view/1969

Proyectos con Arduino. (2022). *Proyectos con Arduino*. Obtenido de Proyectos con Arduino.: https://proyectosconarduino.com/curso/interrupciones-arduino-attachinterrupt/

Romero, V. (19 de enero de 2019). *Youtube*. Obtenido de Youtube: https://www.youtube.com/watch?v=9FW9LiFroKY

Texas instruments. (julio de 2022). Obtenido de Texas instruments: https://www.mouser.mx/ProductDetail/Texas-Instruments/LM311P?qs=0OT4q4QBUTKsKDe9sMZxLw%3D%3D

Universidad de Cantabria. (2016). *Universidad de Cantabria*. Obtenido de Universidad de Cantabria: https://sdei.unican.es/Paginas/servicios/software/Labview.aspx

Anexos

Anexo A programa de Arduino para velocidad y temperatura

```
int NF = 0;
double tiempo = 0;
int cociente = 0;
int detector = 2;
int residuo = 0;
double Temp = 0;


void setup() {
 attachInterrupt(digitalPinToInterrupt(detector),contador,FALLING);
 Serial.begin(9600); // put your setup code here, to run once:
}

void contador ()
{ NF++;
}
```

Fuente creación propia

Figura 51. Parte superior declaración de variables, parte media función de interrupciones y parte baja contador de flancos.

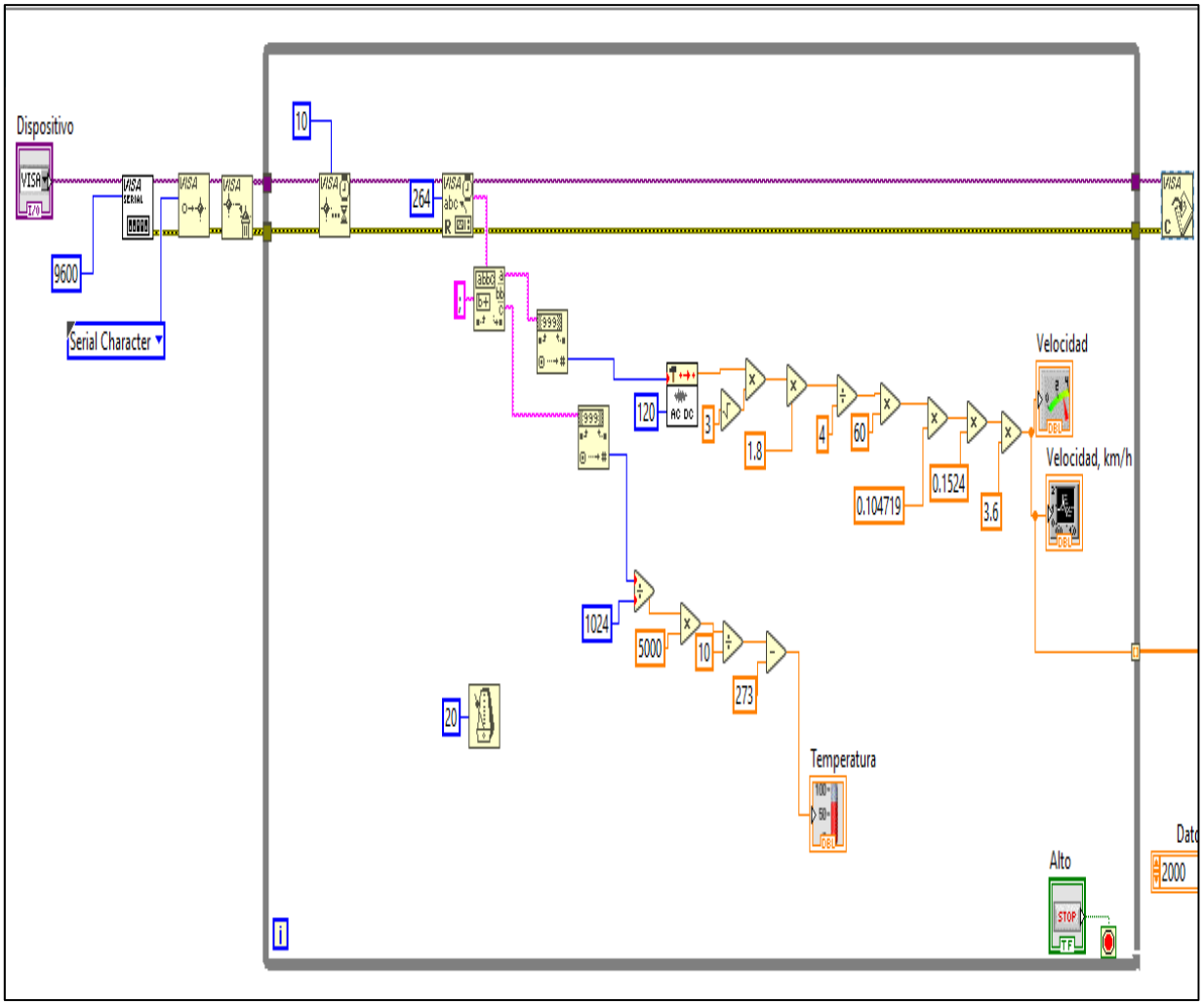
```
void loop() {
  // put your main code here, to run repeatedly:
tiempo = millis();
cociente = (tiempo/1000);
residuo = tiempo -(cociente*1000);
Temp = analogRead(A0);
//NF++;

Serial.print(NF);
Serial.print(";");
Serial.println(Temp);
if (residuo >= 980)
{
NF= 0;
}
delay(20);
```

Fuente creación propia

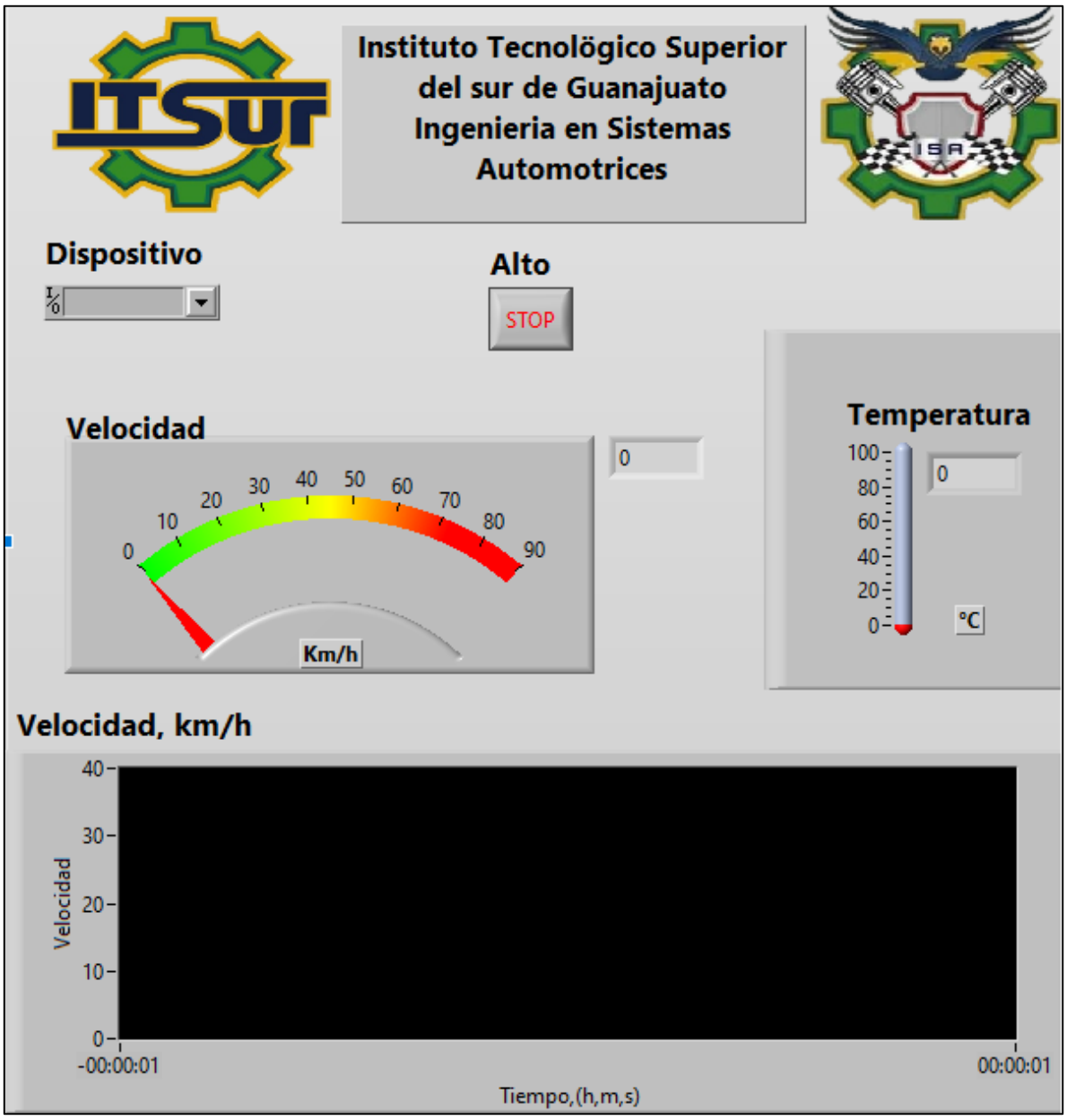
Figura 52. Parte superior operaciones, parte central datos a imprimir y parte inferior condición de reinicio de contador.

Anexo B programa LabVIEW para temperatura y velocidad.



Fuente creación propia

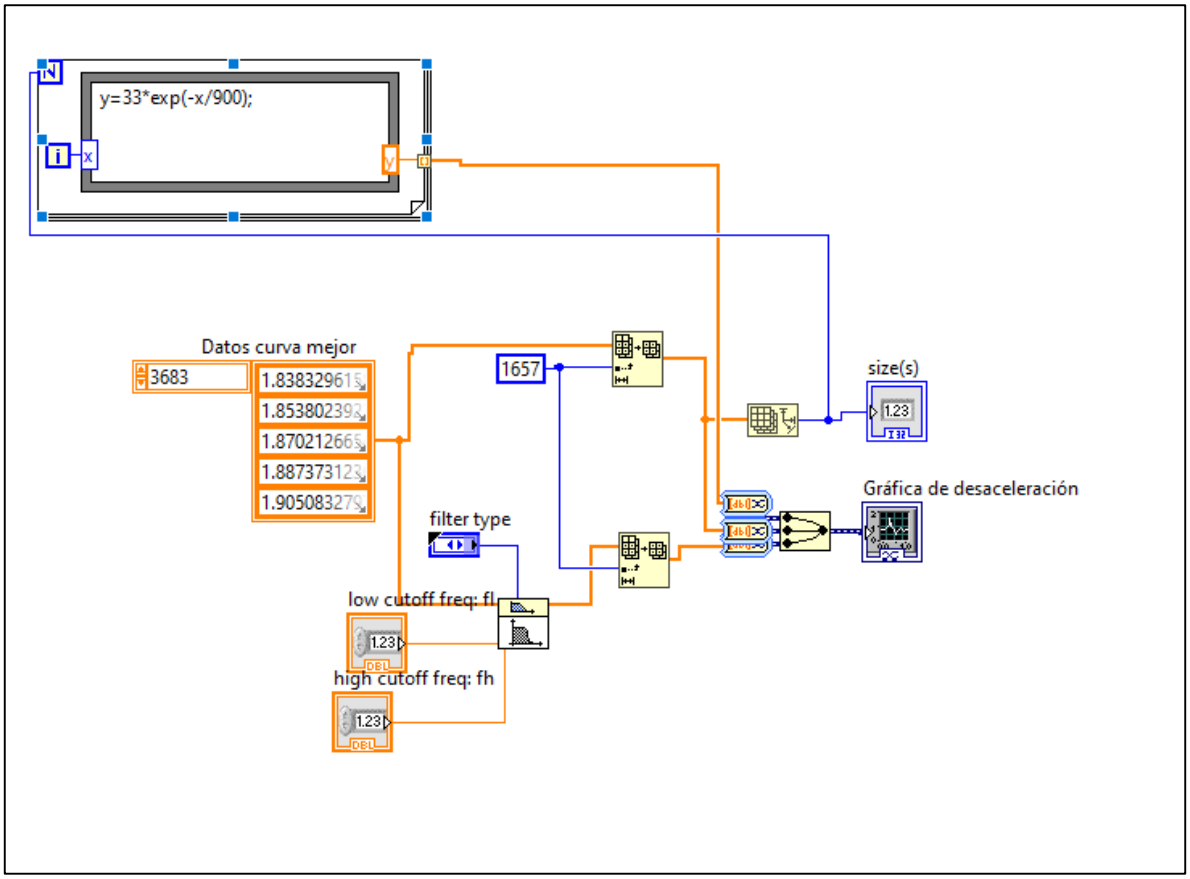
Figura 53. Parte superior instrucción de leer, parte medio procesamiento para velocidad y parte inferior procesamiento para temperatura.



Fuente creación propia

Figura 54.Panel frontal para temperatura y velocidad.

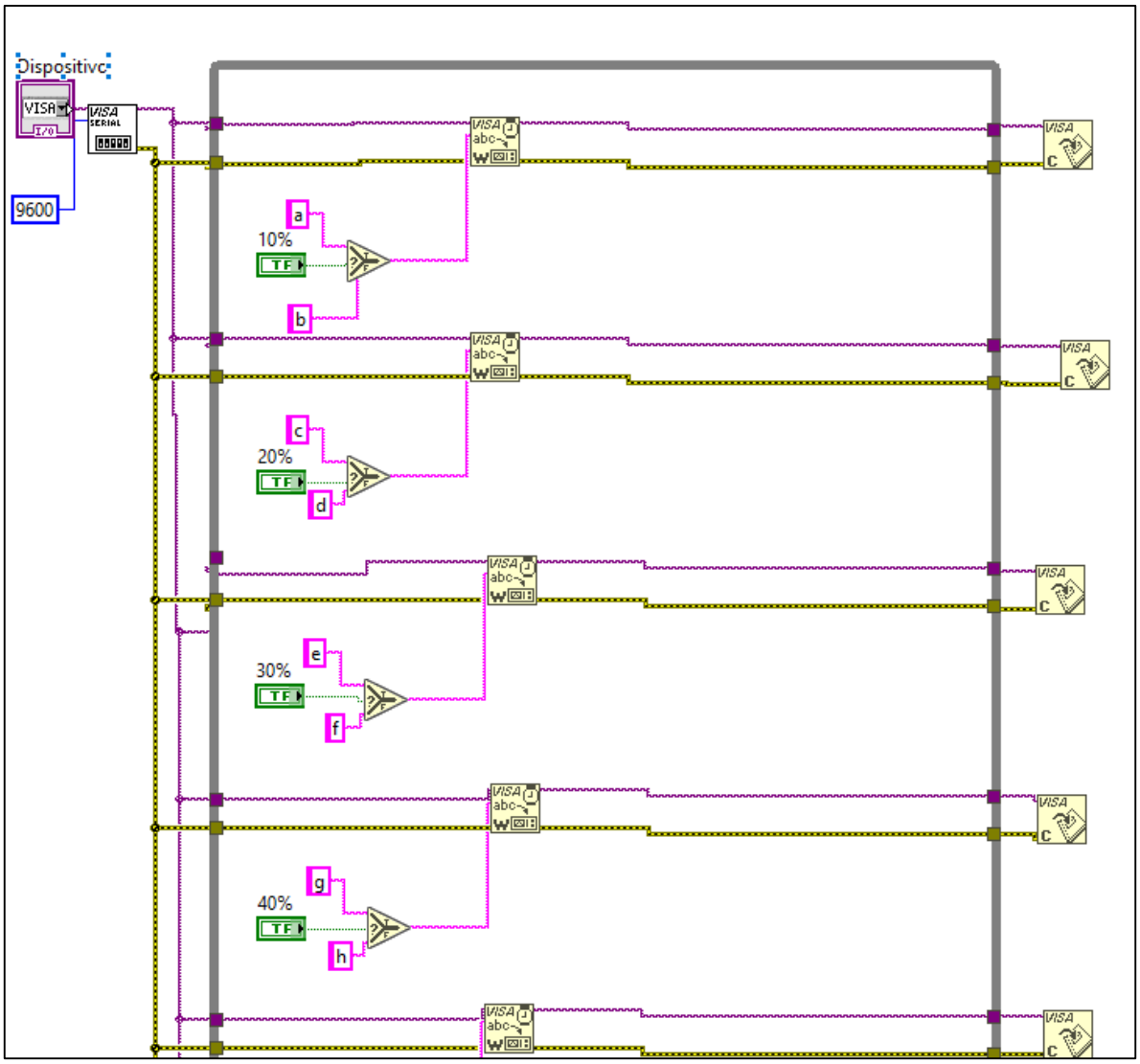
Anexo C programa para graficar desaceleración



Fuente creación propia

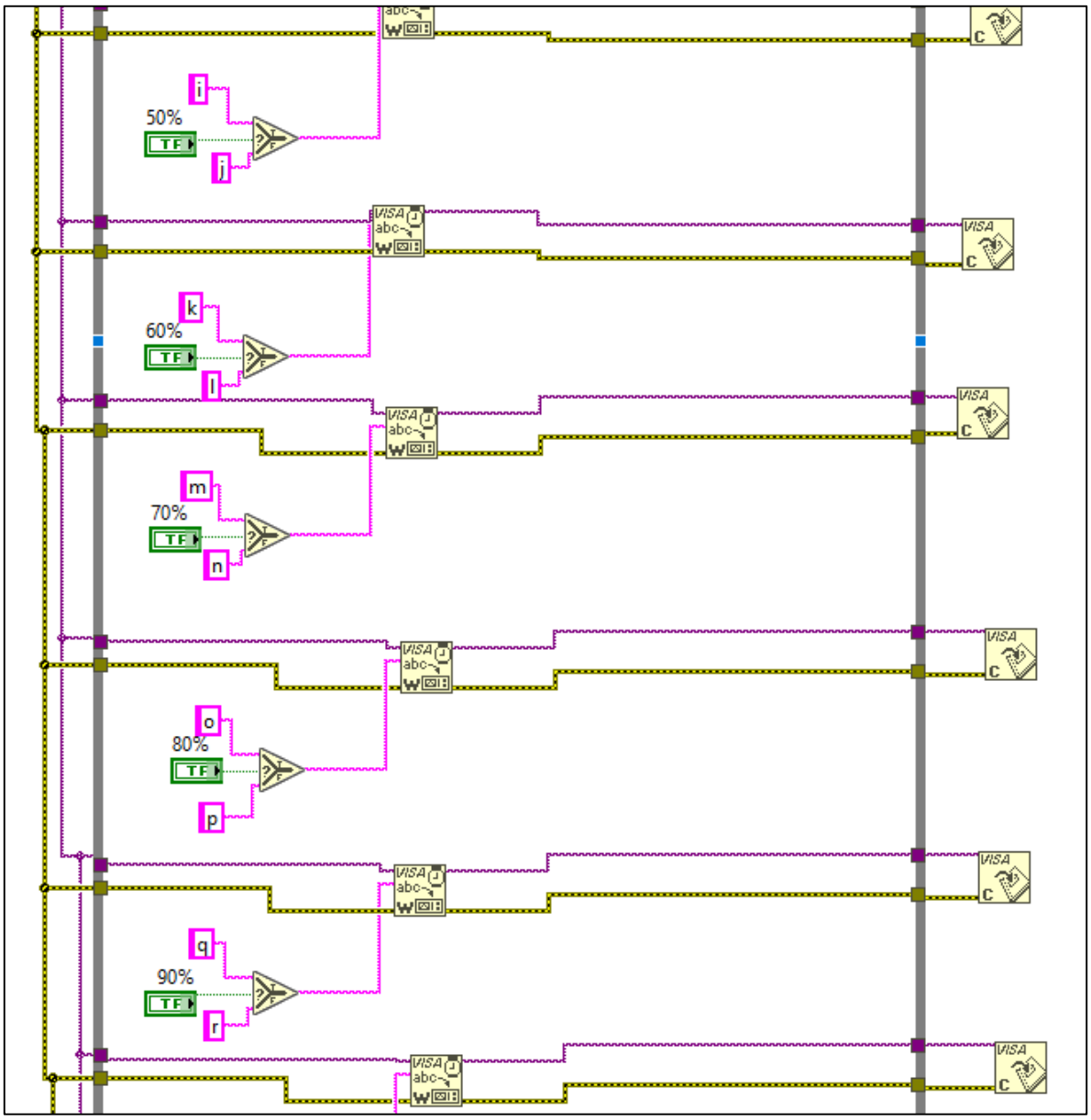
Figura 55. Parte superior ecuación del sistema, parte media grafica real y parte inferior aplicación del filtro.

Anexo D programa de LabVIEW para la válvula



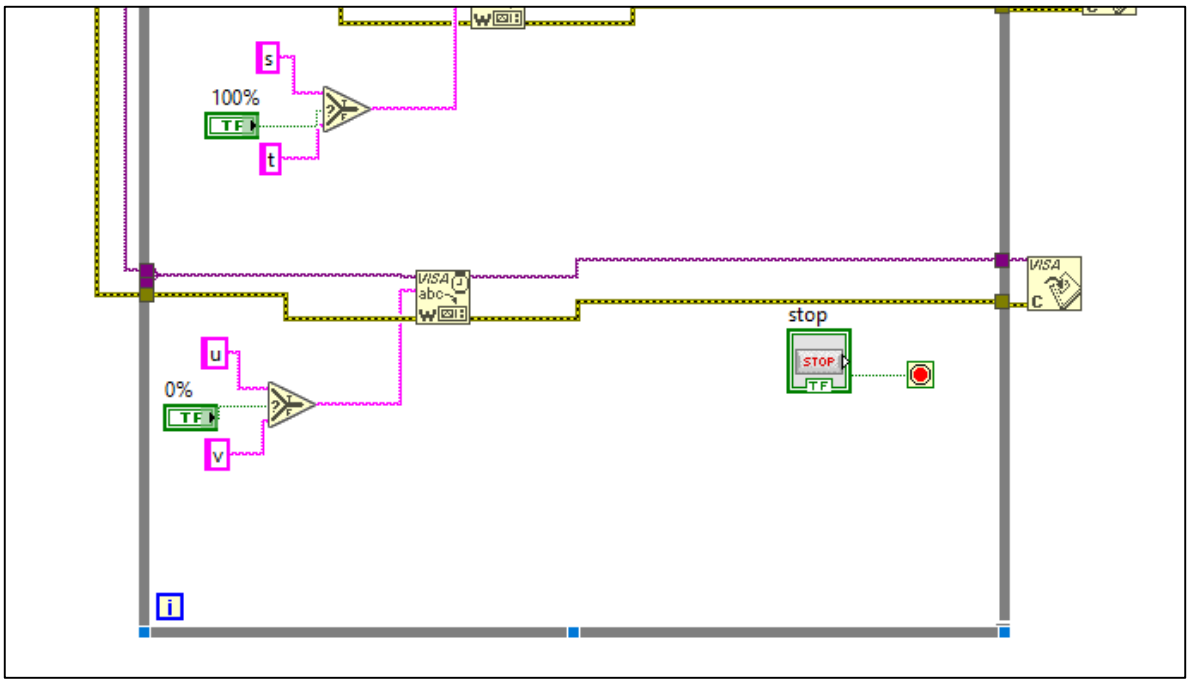
Fuente creación propia

Figura 56. Parte superior configuración para escribir un dato y parte inferior datos a escribir.



Fuente creación propia

Figura 57. Segunda parte de datos a escribir.



Fuente creación propia

Figura 58. Últimos datos a escribir.

Anexo E programa de Arduino para la válvula

```
char valor;
String lectura;

void setup() {

 Serial.begin(9600);
 pinMode (11,OUTPUT); // put your setup code here, to run once:

}

void loop() {
if (Serial.available())
{
  valor = Serial.read();

  if (valor=='a')
  {
    analogWrite(11,26);       //10%
    }

    if (valor=='c')
  {
    analogWrite(11,51);      //20%
    }
    if (valor=='e')
  {
    analogWrite(11,76 );     //30%
    }
    if (valor=='g')
  {
    analogWrite(11,104);     //40%
```

Fuente creación propia

Figura 59. Parte superior declaración de variables y parte inferior casos de modulación PWM.

```
  }
    if (valor=='i')
{
  analogWrite(11,127);    //50%
  }
    if (valor=='k')
{
  analogWrite(11,153);    //60%
  }
    if (valor=='m')
{
  analogWrite(11,178);    //70%
}
    if (valor=='o')
{
  analogWrite(11,204);     //80%
  }
    if (valor=='q')
{
  analogWrite(11,229);     //90%
   }
     if (valor=='s')
{
  analogWrite(11,255);       //100%
  }
    if (valor=='u')
{
  analogWrite(11,0);        //0%
  }

}
```

Fuente creación propia

Figura 60. Complemento de valores de PWM