Simulación y control del equipo de apoyo en tierra para la verificación y chequeo del sistema Rudder Boost de las aeronaves Beechcraft King Air series 200

Presentado por
Juan Carlos Rico Clavijo
Cristhian Fernando Montenegro González

Fundación Universitaria Los Libertadores
Facultad de Ingeniería y Ciencias Básica
Programa de Ingeniería Aeronáutica
Bogotá D.C, Colombia
2020

Página dejada en blanco intencionalmente

Simulación y control del equipo de apoyo en tierra para la verificación y chequeo del sistema Rudder Boost de las aeronaves Beechcraft King Air Series 200

Presentado por
Juan Carlos Rico Clavijo
Cristhian Fernando Montenegro González

En cumplimiento parcial de los requerimientos para optar por el Título de:
Ingeniero Aeronáutico

Dirigida por
Ing. Aixa Ivone Ardila Avellaneda

Codirector
Ing. Pedro Fernando Melo Daza

Presentado a
Programa de Ingeniería Aeronáutica

Bogotá D.C, Colombia
2020

Página dejada en blanco intencionalmente

Nota de Aceptación

_____

_____

_____

_____

_____

Firma del presidente del jurado

_____

Firma del jurado

_____

Firma del jurado

Bogotá D.C, 10 Diciembre 2020

Página dejada en blanco intencionalmente

Las directivas de la fundación Universitaria Los Libertadores, los jurados calificadores y el cuerpo docente no son responsables por los criterios e ideas expuestas en el presente documento. Estos corresponden únicamente a los autores y a los resultados de su trabajo.

Página dejada en blanco intencionalmente

Dedicatoria

Dedicamos este proyecto primeramente a Dios por darnos la oportunidad, paciencia y sabiduría de vivir esta etapa universitaria llena de grandes experiencias y aprendizajes únicos que nos han orientado a descubrir el mundo de la aviación, un mundo lleno de grandes competencias, un campo que se puede admirar por su gran complejidad tecnológica y científica.

A nuestras familias quienes han sido nuestro mayor apoyo en esta carrera de Ingeniería Aeronáutica principalmente nuestros padres, quienes nos han orientado con su gran ejemplo y nos han guiado para ser personas llenas de buenos valores y propósitos en la vida brindándonos su apoyo incondicional siempre en todo momento hasta el final.

A nuestros compañeros de trabajo que con afecto y apoyo moral nos han estado alentando y acompañando.

A quienes del mismo modo fueron nuestro apoyo incondicional como profesores y compañeros de pregrado que de igual forma fueron un envión anímico y motivacional en todo el desarrollo de este proyecto.

Página dejada en blanco intencionalmente

Agradecimientos

En primera instancia, queremos agradecerle a Dios por brindarnos la oportunidad de realizar y desarrollar cada una de las etapas de este proyecto de investigación de forma exitosa, igualmente, agradecemos a cada uno de nuestros profesores de pregrado, quienes nos brindaron sus conocimientos, los cuales fueron de gran ayuda para aplicarlos en las diferentes etapas de nuestro proyecto.

Queremos manifestar nuestros agradecimientos a la decanatura de ingeniería y ciencias básicas, al Ingeniero Jorge luis Nisperuza Toledo Director del programa de Ingeniería Aeronáutica quien nos brindó la oportunidad de conocer y tener alianza con la escuela de aviacion del ejercito nacional, por último, al Ingeniero Pedro Fernando Melo Daza que nos guió con gran parte de este proceso, estuvo muy pendiente de nosotros y fue la persona quien más nos colaboró hasta el último día, brindándonos su total apoyo y conocimiento para así lograr culminar este trabajo satisfactoriamente.

Página dejada en blanco intencionalmente

Índice General

Índice de Figuras

y la válvula proporcional. Fuente: autores.

Figura 3.31 e) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: autores.

Figura 3.31 f) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: autores.

Figura 3.31 g) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: autores.

Abreviaturas

ADC : Conversor Análogo Digital.

AMM : Manual de Mantenimiento de la Aeronave.

ATA : Asociación de transporte Aéreo.

CCW : En sentido contrario a las manecillas del reloj.

CW : En Sentido con las manecillas del reloj.

CISC : Computador de instrucciones complejas.

CPU : Unidad central de procesamiento.

DCO : Oscilador controlado digitalmente.

E/C : Entradas y salidas.

GND : Conexión a tierra.

I : Entero.

MCU : Microcontrolador.

PSI : Libra de fuerza por pulgada cuadrada.

PIN : Terminal.

PWM : Modulación de ancho de pulso.

RISC : Computador de conjunto de instrucciones reducidas.

TI : Texas Instrument.

SCL : Sistema de línea de los pulsos de reloj.

SDA : Línea de datos en serie.

SPI : Bus de interfaz de periféricos.

UART :Transmisor-Receptor Asíncrono.

USB : Bus universal en serie.

GENERALIDADES

Los equipos de apoyo en tierra o bancos de pruebas y ensayos son los encargados de apoyar procesos de carga, suministro de combustible, mantenimiento y logística para todos las necesidades terrestres de las aeronaves, cubren una necesidad muy importante la cual es la optimización de recursos, tiempo y costos muy significativos en el transporte y servicio de estos requerimientos, son la herramienta principal para apoyar la operación de las aeronaves que están en tierra, los equipos pueden ser fijos, móviles o portátiles; su funcionamiento puede ser mecánico, eléctrico o electromecánico y se pueden desempeñar en sistemas eléctricos neumáticos o hidráulicos.

Este proyecto tiene como prioridad simular el equipo de apoyo en tierra mediante el control digital a partir de un lenguaje de programación garantizando el control sobre el sistema y así minimizar procesos reiterativos para luego desarrollar pruebas de chequeo y calibración más efectivas al momento de hacer los servicios de mantenimiento, en la Figura 1.1 (a) y (b) se muestra con exactitud un equipo de apoyo de tierra para el sector aeronáutico para realizar y visualizar pruebas de compresión del motor.

Figura 1.1 (a) Tester de compresion portátil para la aeronáutica - Bauer inc USA Connecticut. [26]

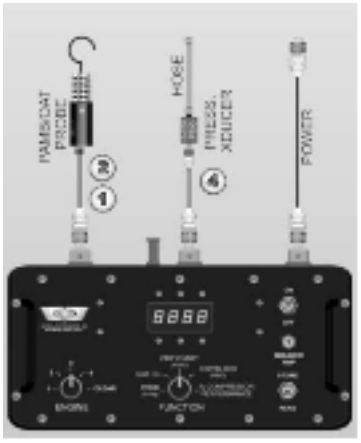

Figura 1.1 (b) Tester de compresion portátil para la aeronáutica - Howell instrument, Inc. [26]

Los equipos de apoyo en tierra pueden ser de uso general o particular, es decir puede ser compatibles para realizar pruebas en diferentes tipos de aeronaves o también ser de uso exclusivo de un solo tipo de aeronave, el equipo que se va a simular va orientado únicamente a pruebas neumáticas para aeronaves King Air Series 200, la cual se puede visualizar en en la Figura 1.2.

Figura 1.2 Avión Tipo Beechcraft King Air del Centro de Investigación de Vuelo Dryden de la NASA. [21]

Este equipo tiene cómo finalidad realizar pruebas neumáticas en tierra en las líneas de presión del estabilizador vertical de la aeronave, a partir de este equipo se pretende generar una presión inicial, utilizando una pipeta de nitrógeno para el correcto funcionamiento del equipo. Claramente se deben seguir las recomendaciones del fabricante, en la Figura 1.3 se pueden visualizar el timón de dirección y de profundidad de la aeronave después de realizar una prueba de mantenimiento, que tiene cómo único fin, verificar si estos sistemas de control principal están funcionando de una forma correcta.

Figura 1.3 Timón de dirección y de profundidad de la aeronave King Air Series 200 posicionados y con la prueba de chequeo realizada según las instrucciones del fabricante [26]

Adicionalmente, este equipo sirve de uso general y se puede adecuar en pruebas de otras aeronaves y superficies de control, solamente si las instrucciones del fabricante indican que las capacidades del equipo cumplen con el intervalo de presión a suministrar

RESUMEN

Por parte de la Escuela de Aviación del Ejército Nacional de Colombia ESAVE, surgió la necesidad de simular un equipo de apoyo en tierra, para verificar el sistema Rudder Boost de las aeronaves King Air Series 200, a través de un lenguaje de programación claro y conciso mostrando funciones claras y óptimas desde un interfaz gráfico, donde su único fin fue integrar un sistema de control digital a partir de un principio de operación neumático, y así lograr llevar a cabo una serie de funciones de operación y pruebas para cumplir con las necesidades requeridas y propuestas en esta simulación

Se realizó un caso de estudio sobre los diferentes tipos de Equipos de apoyo en tierra esenciales al momento de realizar el mantenimiento de estas aeronaves, se analizó el funcionamiento del sistema rudder boost, costo beneficio y el procedimiento que se debe llevar a cabo para la verificación y el chequeo según las instrucciones del fabricante, se estudiaron los conceptos previos que fueron fundamentales para la óptima situación de control del equipo de apoyo en tierra, dentro de los cuales se encontraron los principios de funcionamiento de los componentes electroneumáticos.

A partir de los conceptos previos, generalidades y principios de funcionamiento se logró entender cómo simular el equipo de apoyo en tierra, dentro del mantenimiento general, interiorizando las pruebas y verificación del sistema rudder boost, mediante los reportes de mal funcionamiento, reportes operacionales, demoras en la ejecución de las pruebas y verificaciones, y el desajuste y mal uso continuo de la herramienta, permitió idear una simulación a partir de todas esta recolección de datos mantenimiento, que de una u otra forma se vieron con mayor detenimiento en el dia a dia debido a la operatividad de la misma, con ello se generó un esquemático en el cual se involucraron los pasos jerárquicos para así desarrollar a partir de un interfaz gráfico llamado Processing las aplicaciones necesarias por medio de un código de lenguaje dentro del software y para esto se necesitó gran parte de tiempo y espacio para apropiar el alcance de la herramienta y así dentro de un algoritmo se logró obtener el resultado esperado para un buen desempeño en la simulación del equipo de apoyo en tierra.

Finalmente, se comprobó el funcionamiento del equipo y cada uno de sus componentes electroneumáticos por medio de pruebas que se realizaron en simultáneo en los softwares Processing, arduino y Energía; para la visualización del panel principal y la programación del equipo respectivamente, y así obtener los resultados que garantizaron los parámetros sugeridos por el fabricante.

PALABRAS CLAVES: *Equipo de apoyo en tierra, timón de dirección, sistema rudder Boost, presión neumática, sistema digital, programación, microcontrolador y electromecánica.*

ABSTRACT

As part of the Colombian army aviation school ESAVE, we have determine the need to simulate the GSE rudder boost system, for the king Air 200 Series aircraft, through a concise programming language showing clear and optical functions from a graphical interface, in which its sole purpose is to integrate a digital control system from the beginning of pneumatic operation and achieve a series of operating and testing functions to meet the required and proposed needs in this simulation

A case of study was carried out on the different types of essential ground support equipment when performing maintenance on these aircrafts and described as follows, the operation of the rudder boost system, a cost-benefit relation, and the procedure to be carried out as a verification, were analyzed and a follow up according to the manufacturer's instructions. The previous concepts that were fundamental for the optimal control situation of the ground support equipment were studied, within the operating principles of the pneumatic and electro pneumatic components.

From the previous concepts, generalities and operating principles, it was possible to understand how to simulate the ground support equipment, within general maintenance, internalizing the tests and verification of the rudder boost system, through reports of malfunction, operational reports, delays in the execution of the tests and verifications, the misalignment and continuous misuse of the tool; allowed us to devise a simulation based on all this collection of maintenance data, which in one way or another was seen in greater detail on a daily basis due to its operation. When this schematic was generated the hierarchical steps were involved in order to develop from a graphical interface called Processing, the necessary applications through a language code within the energy software and for this, it took a lot of time and space to appropriate the scope of the tool and within an algorithm was achieved to obtain the expected result for a good performance in the optimization of the ground support equipment.

Finally, the operation of the equipment and each of its electro pneumatics components was verified by means of tests that were carried out simultaneously in the Processing , Arduino and Energy softwares, for the display of the main panel and the programming of the equipment respectively, and thus obtained the results that guaranteed the parameters suggested by the manufacturer

KEYWORDS: *Ground support equipment, rudder, rudder boost system, pneumatic pressure, digital system, programming, microcontroller y electromechanics*

Capítulo 1

INTRODUCCIÓN

"Medir el proceso de un proyecto de programación por líneas de código, es cómo medir la construcción de un aeroplano por su peso." Bill Gates.

El mantenimiento en el sector aeronáutico es un conjunto de procedimientos que permite asegurar que una aeronave, motor, hélice o pieza cumpla con los requisitos aplicables de aeronavegabilidad y se mantengan en condiciones de operación de modo seguro durante toda su vida útil.

Los altos niveles de seguridad en la aviación civil se han alcanzado en gran medida por el cuidadoso diseño exigido a los fabricantes de aeronaves, así como al estricto mantenimiento posterior que garantiza que el avión es fiable en todo momento. Un numeroso grupo de ingenieros y técnicos de mantenimiento trabajan diariamente para garantizar el correcto funcionamiento de las aeronaves. Se trata de trabajos especializados que requieren una acreditación según alcance y son regulados según su estado de matrícula *[32]*

A Partir de estas condiciones, la Fundación Universitaria los Libertadores y Escuela de Aviación del Ejército Nacional de Colombia ESAVE han realizado mejoras y avances en los procedimientos de mantenimiento por medio de investigaciones e innovaciones con el fin de mejorar equipos de apoyo en tierra que sirvan para realizar uno o varios procedimientos de mantenimiento con la mayor eficiencia posible, es decir que lo que se busca es suplir la necesidad de optimizar costos, tiempo y horas hombre, que es lo que las empresas más necesitan para poder generar un mayor impacto en la industria y en la economía del sector.

El desarrollo del presente proyecto proviene de una alianza entre la Fundación Universitaria los Libertadores y la Escuela de Aviación del Ejército Nacional de Colombia ESAVE, en el que se pretende simular un equipo de apoyo en tierra para contribuir en la reducción de costos, tiempo y horas hombre en algunos procedimientos de mantenimiento de los aviones King Air Series 200 de la flota aérea del Ejército Nacional de Colombia.

La investigación del proyecto estará enfocada en simular el equipo apoyo en tierra con el fin de que este este sea operado digitalmente y no manualmente, este se encargará de inyectar nitrógeno a diferentes niveles de presión en algunos sistemas neumáticos que generan el movimiento de las superficies de control del avión , principalmente del timón de dirección y así verificando su correcto funcionamiento.

Su operación y control se dará directamente desde una computadora que estará conectado a una tarjeta controladora la cual estará programada para recibir todas las

instrucciones dadas y emitirlas a los componentes electroneumáticos, por ende, tanto el control del equipo como la tarjeta y los componentes tendrán comunicación electrónica por medio de circuitos que interactúan entre sí.

Con este proyecto se busca aplicar los conocimientos recibidos a lo largo de la carrera de Ingeniería Aeronáutica, muchos de los equipos de apoyo en tierra en la industria funcionan manualmente, así que con este documento se podrá aportar a posibles mejoras similares de otros equipos de apoyo en tierra para que haya un mayor desempeño y competitividad en los campos de la investigación y desarrollo de la escuela de aviación militar ESAVE, fundación universitaria los libertadores y el avance tecnológico de la aviación nacional colombiana.

Capítulo 2

PLANTEAMIENTO DEL PROBLEMA

¿cuales son las condiciones y parámetros de operación que deben tener el equipo de apoyo en tierra en cada uno de sus componentes para que sus pruebas sean precisas, confiables y que se ajuste a las necesidades del servicio de mantenimiento?.

El equipo de apoyo en el tierra fabricado por los estudiantes de la Escuela de Aviación del Ejército Nacional de Colombia ESAVE se opera de forma manual, es decir que la presión a suministrar se gradúa por medio de perillas de giro, el aumento de la presión se visualiza por el recorrido de la aguja en los manómetros, es importante buscar una opción que tenga mayor facilidad en su operación por lo tanto, se propone realizar una simulación la cual consiste realizarla en un equipo de apoyo en tierra para resolver los distintos contratiempos y demora de operación que presenta este modelo inicial y así hacer que funcione digitalmente y no manualmente, esto garantiza mayor facilidad y desempeño en la operación del equipo.

El banco de pruebas a simular propone tener una capacidad de operar de tal manera que se logra tener bajo costo de combustible y minimizar tiempos de operación a la hora de realizar verificaciones y chequeos cuando los servicios de mantenimiento lo requieran, a través de la simulación propuesta se desea obtener con exactitud la fuerza sobre unidad de área que actúa en las líneas neumáticas conectadas al timón de dirección garantizando los parámetros sugeridos por el fabricante en todo momento desde su ejecución hasta la verificación total del sistema.

2.1 Objetivos

2.1.1 Objetivo General:

Simular la electrónica y el control de operación de un equipo de apoyo en tierra para la verificación y chequeo del sistema Rudder Boost de las aeronaves King Air Series 200 establecido por el fabricante.

2.1.2 Objetivos Específicos:

- Entender el funcionamiento neumático, parámetros establecidos por el fabricante, funcionalidad principal del sistema Rudder Boost de la aeronave King Air Series 200.
- Establecer el diagrama general del equipo de apoyo en tierra, el algoritmo de funcionamiento y operación, que muestre la interacción entre los componentes electroneumáticos seleccionados y las tarjetas controladoras propuestas.
- Programar las tarjetas controladoras por medio de un código de lenguaje en los softwares Arduino y/o Energía que cumpla con las condiciones de funcionamiento y operación de los componentes del equipo.
- Ilustrar la operación del equipo a través de un panel de control en el software processing por medio de una comunicación con arduino y/o energía para la ejecución del código programado.

2.2 Justificación

Está propuesta es una necesidad que inicia en el año 2016 donde se implementa un equipo de apoyo en tierra por parte de los estudiantes de la Escuela de Aviación del Ejército Nacional de Colombia ESAVE para realizar verificación y chequeo del sistema rudder boost de la flota de aeronaves King Air Series 200 del Ejército Nacional de Colombia, estas pruebas consisten en la inyección de cierta cantidad de aire y/o nitrógeno a presión en los conductos neumáticos del sistema, cumpliendo así con la labor de mantenimiento para verificar si el sistema rudder boost funciona correctamente por medio del movimiento del timón de dirección, sin embargo este equipo funciona manualmente y no está automatizado, por lo tanto es necesario diseñar un equipo que se adapte a una operación automatizada y con principios electroneumáticos dirigida desde un panel principal de operación.

De acuerdo con lo anteriormente mencionado, en el batallón de aviación No. 1 del ejército se presenta la necesidad de realizar una simulación del equipo de apoyo en tierra automatizado para realizar el servicio de mantenimiento mencionado anteriormente garantizado el eficaz cumplimiento de las pruebas de chequeo y funcionamiento del sistema Rudder Boost , además se necesita minimizar el consumo de combustible en la prueba, ya que sin este equipo, los motores deben encenderse, pero con el equipo los motores pueden estar apagados, es evidente que también se busca reducir costos y tiempo de servicio en esta prueba de mantenimiento.

Con este equipo, se busca que la prueba de mantenimiento se realice con mayor facilidad, ya que al ser diseñado con principios electroneumáticos su operación evita la graduación manual de la presión, ahorrando tiempo y optimizando procesos establecidos en el manual de mantenimiento. Una de las mayores ventajas de usar equipos electroneumáticos automatizados es que los resultados tienen mayor precisión a comparación de los resultados adquiridos en los equipos operados manualmente, por esta razón se busca que el equipo de apoyo en tierra propuesto funcione de forma electroneumática.

Como ya se sabe el mantenimiento es de vital importancia para la industria aeronáutica debido a que garantiza que el funcionamiento y operación de los sistemas de las aeronaves se realicen de forma correcta y segura, teniendo confiabilidad y generando el crecimiento tecnológico adecuado para la producción del crecimiento de la industria aeronáutica de manera exponencial donde la optimización de los procesos de mantenimiento con ayuda de estos equipos de apoyo, sean de gran impacto en el sector Aeronáutico del país.[32]

2.3 Alcance del proyecto

2.3.1 Observaciones

En el sector Aéreo la filosofía de mantenimiento ha evolucionado de gran forma y garantiza un amplio espectro de conocimiento relativo en los procesos de ingeniería, uno de estos hace referencia a los procesos electroneumáticos y de programación que en gran parte constituye la simulación y control del equipo de apoyo en tierra para la verificación y chequeo del sistema Rudder Boost del Beechcraft King Air Series 200 propuesto

El equipo a desarrollar está conformado por dos Fases, en la Fase 1 se pretende realizar el control del equipo propuesto complementado con una simulación del sistema eléctrico del mismo y en la Fase 2 se requiere realizar el diseño conceptual preliminar para la fabricación del equipo se 2.

El presente proyecto tiene cómo alcance únicamente solo el desarrollo de la fase 1, con el fin de establecer una base sólida del programa de control del equipo de apoyo en tierra para abrir el campo de conocimiento no solo para el desarrollo de la Fase 2, sino también para complementar el control de equipos que funcione con los mismos principios electromecánicos y de automatización electrónica.

Cabe resaltar que por el tema de la emergencia sanitaria frente a la pandemia por COVID-19 y el aislamiento obligatorio esta fase 2 no se logró finalizar.

2.3.2 Fase 1 - Simulación y control del equipo

Con base en el funcionamiento del sistema Rudder Boost de la aeronave King Air Series 200, se establece un diagrama general de un equipo de apoyo en tierra que tiene cómo fin realizar una prueba de chequeo con principios funcionales neumáticos, el equipo debe contar con un sistema electroneumático para su control, a su vez requiere de una tarjeta controladora que se encargue de activar los componentes eléctricos y electroneumáticos por medio de señales electrónicas para su operación.

La simulación de control busca establecer los parametros principales que debe tener todo el sistema eléctrico y los tipos de componentes que conforman el equipo, para luego programar bajo esta información un algoritmo de control que cumpla con las condiciones de funcionamiento y operación estimadas partiendo de las sugerencias establecidas por el fabricante y así mostrar una simulación del equipo, luego determinar qué tan viable es y si a futuro se pueda seguir mejorando de una forma exponencial este tipo de proyectos que enriquece el sector aéreo.

2.3.3 Fase 2 - Diseño y fabricación del equipo

Para determinar el alcance final de la fase 2 de este proyecto, es necesario establecer el funcionamiento general del equipo y realizar los estudios estructurales, neumaticos y electricos, ya que el diseño y la fabricación de este equipo requiere de estos complementos, además de un diseño conceptual previamente establecido; los estudios estructurales son necesarios para determinar el diseño apropiado a partir de los material seleccionados correctamente.

Dentro de su configuración conceptual este equipo de apoyo en tierra poseerá una carcasa portátil en la cual se mantendrán almacenado los componentes y con ello se cumplira con el proceso de preservaciòn que es de vital importancia en el área de mantenimiento.

Por último los estudios de electrónica y electroneumática son necesarios al momento de continuar con esta fase de diseño y fabricación, aparte de esto también es necesario tener conocimientos previos de programación y sistemas de control.

Las condiciones del sistema neumático implican la capacidad de los componentes de soportar las magnitudes de presión de operación, luego de ello debe ensamblarse con el sistema electrónico desarrollado inicialmente en la fase 1 del proyecto, finalmente se debe ensamblar el equipo en su totalidad con todos los detalles visuales, las pruebas a realizar del funcionamiento del equipo se realizarán en el proceso de chequeo del avión.

Capítulo 3

MARCO TEÓRICO

3.1 CONTROL DE VUELO PARA LOS AVIONES KING AIR SERIES 200

3.1.1 Superficies De Control

Las superficies de control principales de una aeronave son los alerones, los elevadores y el timón de dirección, son controladas por el piloto con ayuda del volante y los pedales que se pueden observar en la Figura 3.1.

Figura 3.1 Controles de vuelo de las aeronaves King Air Series 200. [26]

Aparte de los flaps como superficies de control existen unas aletas compensadoras adicionales que son llamadas trim tabs , estas ayudan a neutralizar las vibraciones las cual

permite mantener una mejor estabilidad y control de la aeronave.

Cabe resaltar que los Flaps son superficies secundarias adicionales en las alas del avión, sirven para mejorar la sustentación y mantener un vuelo más controlado a menor velocidad. La Figura 3.2 muestra la distribución y posición de las superficies de control en el avión.

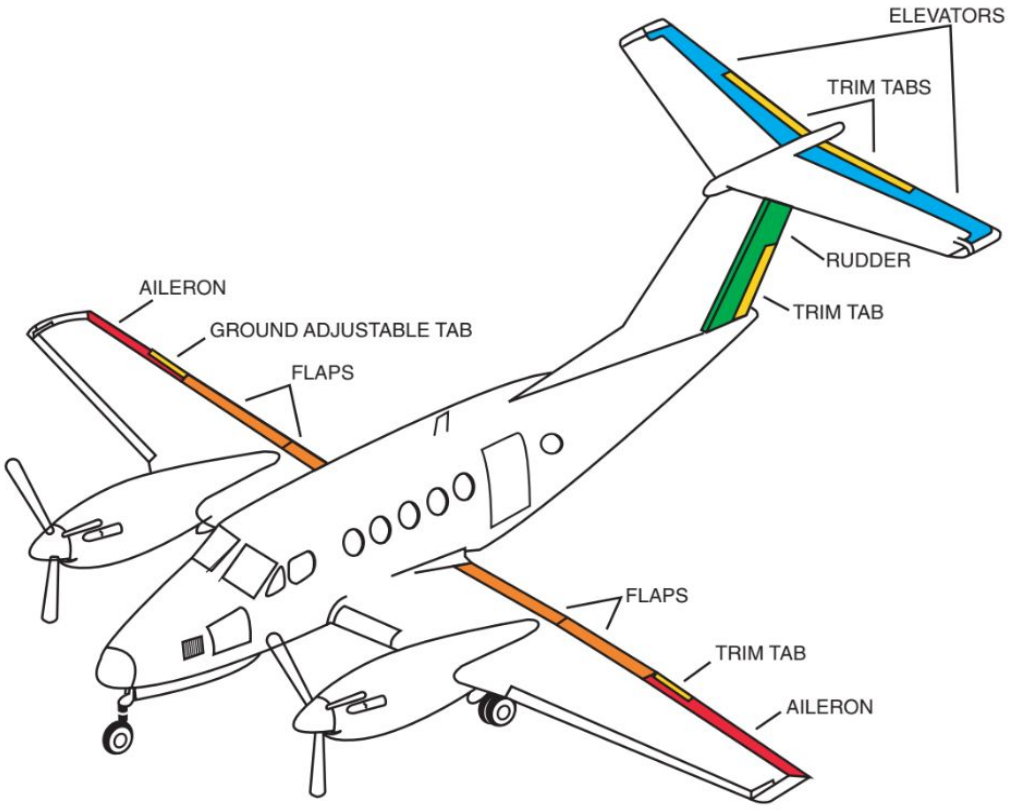

Figura 3.2 Diagrama de las superficies de control de las aeronaves King Air Series 200. [26]

Los sistemas que utilizan las superficies de control para su movimiento son mecánicas a partir de guayas que funcionan mediante control electrónico y a partir de motores lo cual mueven las superficies y así cumplen con el objetivo.

- **Alerones.** Son las superficies de control que están posicionadas en los extremos de las alas, el piloto las controla por medio del volante de la cabina. [26]

  Si el volante gira hacia la derecha, el alerón derecho sube para disminuir la sustentación y el izquierdo baja para aumentar la sustentación, si el volante gira a la izquierda, ocurre lo contrario, con esta operación se busca realizar un viraje en la aeronave en ambos sentidos [26].

- **Elevadores.** Son las superficies que se encuentran en el estabilizador horizontal del avión, se controlan ejerciendo presión hacia al frente en el volante o tirando del volante hacia atrás.[26]

Si se tira del volante, tomando como referencia la vista del perfil izquierdo del avión, los elevadores suben, el combustible en los tanques interactúa en los elevadores generando una sustentación y un momento en sentido con las manecillas del reloj "CW" con respecto al eje transversal del avión , de esta forma se aumenta el ángulo de ataque de las alas, se gana sustentación y ascenso de la aeronave.

Si se ejerce presión en el volante ocurre lo contrario, los elevadores bajan, el combustible interactúa con los elevadores generando una fuerza y un momento en sentido contrario a las manecillas del reloj " CCW" , con respecto al eje transversal del avión, de esta forma se disminuye el ángulo de ataque de las alas, se pierde sustentación y la aeronave desciende.

- **Timón de dirección.** Es la superficie que se ubica en el estabilizador vertical, es controlada por los pedales de la cabina que acciona el piloto. [26]

Si se acciona el pedal derecho, tomando como referencia la vista superior del avión, el timón se flecta hacia la derecha, el curso de la aeronave empieza a cambiar hacia la derecha a causa del momento en sentido con las manecillas del reloj "CW", generado con respecto al eje vertical.

Si se acciona el pedal izquierdo, el timón se flecta hacia la izquierda y el curso de la aeronave empieza a cambiar hacia la izquierda a causa del momento en sentido contrario a las manecillas del reloj " CCW" , generado con respecto al eje vertical.

3.1.2 Ejes de Rotación y Maniobras Para La Estabilidad del avión

---

El movimiento de las superficies de control permite que la aeronave realice maniobras de giro a través de los ejes transversal, longitudinal y vertical, los cuales son los principales ejes que mantienen a la aeronave en su estructura ya que estos se unen en el centro de gravedad del avión.

En estos ejes de referencia se realizan las maniobras de cabeceo, alabeo, y guiñada establecidos como los movimientos de maniobra principales que hace un avión en vuelo para direccionar su desplazamiento controlado, en la Figura 3.3 se evidencia un diagrama de una aeronave con sus ejes de referencia y los movimientos de maniobra que debe desempeñar la aeronave en cada eje.

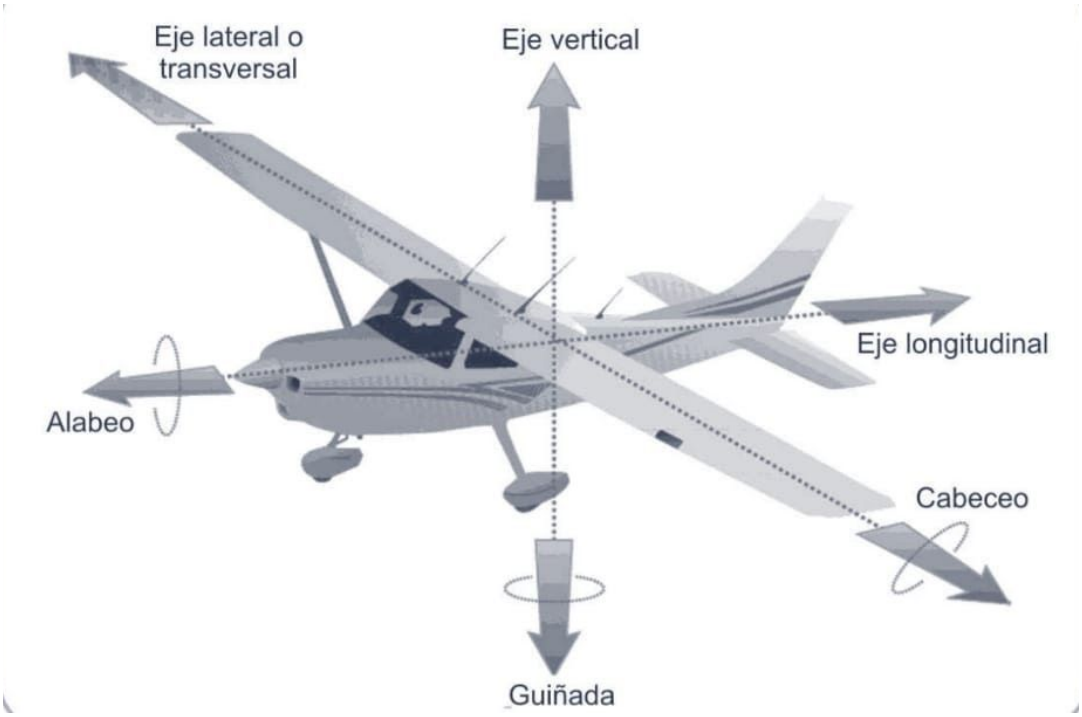

Figura 3.3 Ejes de referencia en la aeronave con los movimientos básicos que debe realizar en cada uno, para un desplazamiento estable y controlado. [26]

- **Eje longitudinal.** Es el eje de referencia a lo largo de la aeronave posicionado de forma simétrica y centrada, es decir que este va desde el empenaje hasta la punta de la nariz del avión y pasa por el centro de gravedad de la aeronave.

- **Eje vertical.** Eje de referencia perpendicular al plano de vista superior e inferior de la aeronave, que pasa por el centro de gravedad.

- **Eje transversal.** El eje de referencia que pasa por el centro de gravedad y va paralelo al eje de punta a punta de las alas, se posiciona perpendicular al plano de vista de perfil izquierdo o derecho de la aeronave.

- **Alabeo.** Es la maniobra en la cual la aeronave gira con respecto al eje longitudinal para direccionar la aeronave a la derecha o izquierda formando una curva en la trayectoria, las superficies de control que se usan para esta maniobra son los alerones.

- **Guiñada.** Es la maniobra que la aeronave realiza para girar con respecto al eje vertical, con el fin de cambiar la dirección del trayecto vertical de la aeronave hacia la derecha o izquierda, puede realizarse junto con el alabeo si es necesario para compensar cargas aerodinámicas, la superficie de control que se usa para esta maniobra es el timon de direccion .

- **Cabeceo.** En esta maniobra la aeronave gira en referencia al eje trasversal con el fin

de descender o ascender, direccionando la nariz del avión con un ángulo de ataque positivo no mayor al ángulo de ataque máximo establecido en la aeronave según especificaciones del fabricante, o con un ángulo de ataque negativo que no supere el ángulo máximo que de acuerdo al certificado tipo King Air Series 200 no debe ser mayor a 20 grados, las superficies de control que se usan para esta maniobra son los elevadores.

3.2 SISTEMA RUDDER BOOST DE LOS AVIONES KING AIR SERIES 200

3.2.1 Descripción general del sistema

Es un sistema llamado Rudder Boost que tiene la aeronave King Air Series 200 , su función principal es ayudar a controlar la estabilidad lateral a través del timón de dirección en caso de falla, por ejemplo en pérdida de potencia de los dos motores en vuelo este sistema se activa inmediante, en función de la etapa de compresión de alta, dependiendo del motor que falle el motor inverso compensará esta presión mediante un ducto de sangrado que va interconectado directamente al sistema Rudder Boost el cual tiene un rango de operación de 90 a 120 psi de presión neumática, su objetivo es ayudar al timón de dirección a compensar esta inestabilidad y a igualar este empuje requerido para esta aeronave, por esta razón este sistema es de vital importancia para garantizar la seguridad operacional en cualquier fase de vuelo .[26]

3.2.2 Operación del Sistema

Para activar el funcionamiento del sistema Rudder Boost primero se debe colocar el interruptor en la posición Rudder Boost **encendido (ON)**, este interruptor se encuentra en el pedestal de la cabina que se muestra en la Figura 3.4 y los interruptores de las **válvulas de sangrado de aire (Bleed Air Valve)** se colocan en posición **abierta (Open)** o en la posición **Envir off,** [26] Las cuales se encuentran en el panel de control que se muestra en la Figura 3.5.

Por medio de un **interruptor de presión diferencial** llamado **(Delta P)**, se determina la presión de sangrado de cada motor, si la diferencia de presión entre los motores se encuentra dentro de los **(60 ± 5 PSI)** el circuito se cierra para activar la apertura de una válvula solenoide que proporciona sangrado de aire a presión dirigido al servo del Rudder

Boost, el cual se encarga de impulsar el timón de dirección para proporcionar una mejor estabilidad direccional, en la Figura 3.6 se puede evidenciar el diagrama del sistema y la interacción de sus componentes. [26]

Para garantizar que el sistema quede desconectado se debe colocar el interruptor del control de las válvulas izquierda o derecha en la posición **INSTR & ENVIR OFF**, [26] que se muestra en la Figura 3.5.

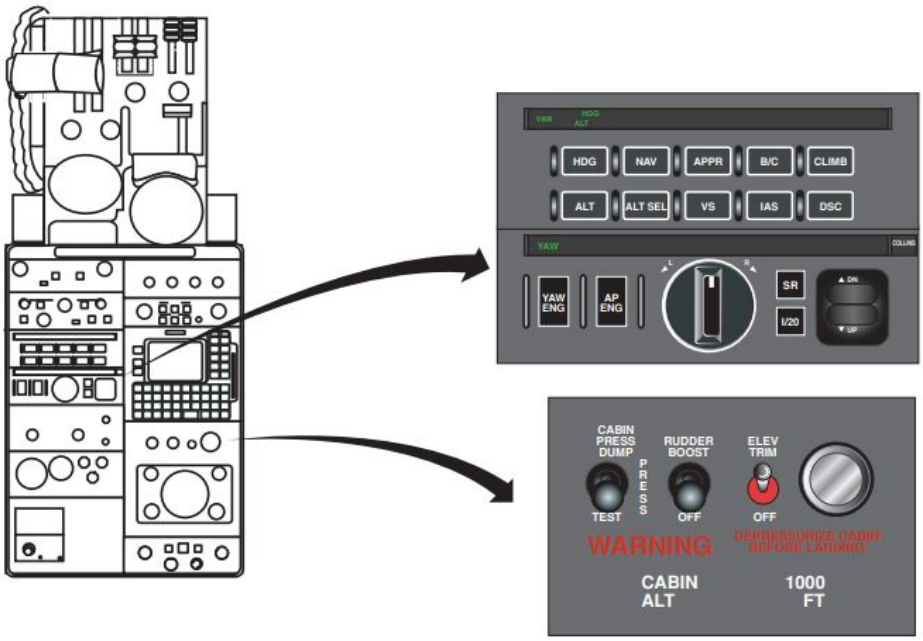

Figura 3.4 . Pedestal donde se activa el sistema Rudder Boost. [26]

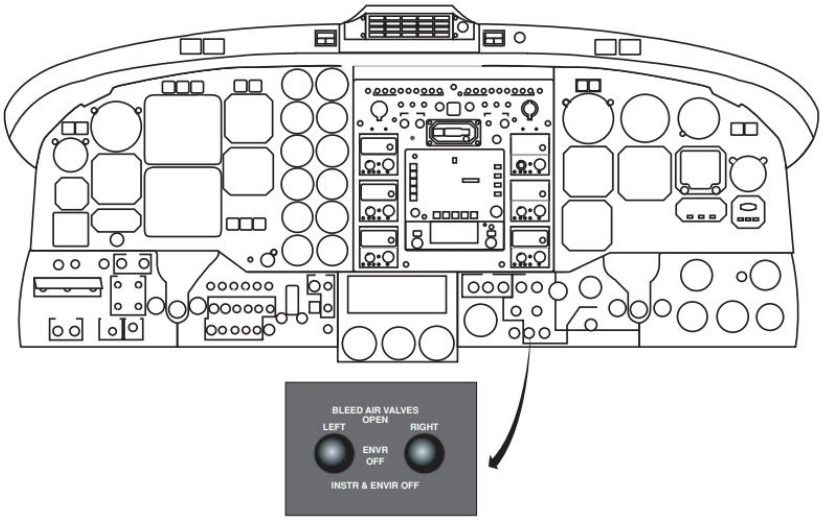

Figura 3.5 Panel del control de las válvulas de sangrado de aire del motor que suministran aire al sistema Rudder Boost. [26]

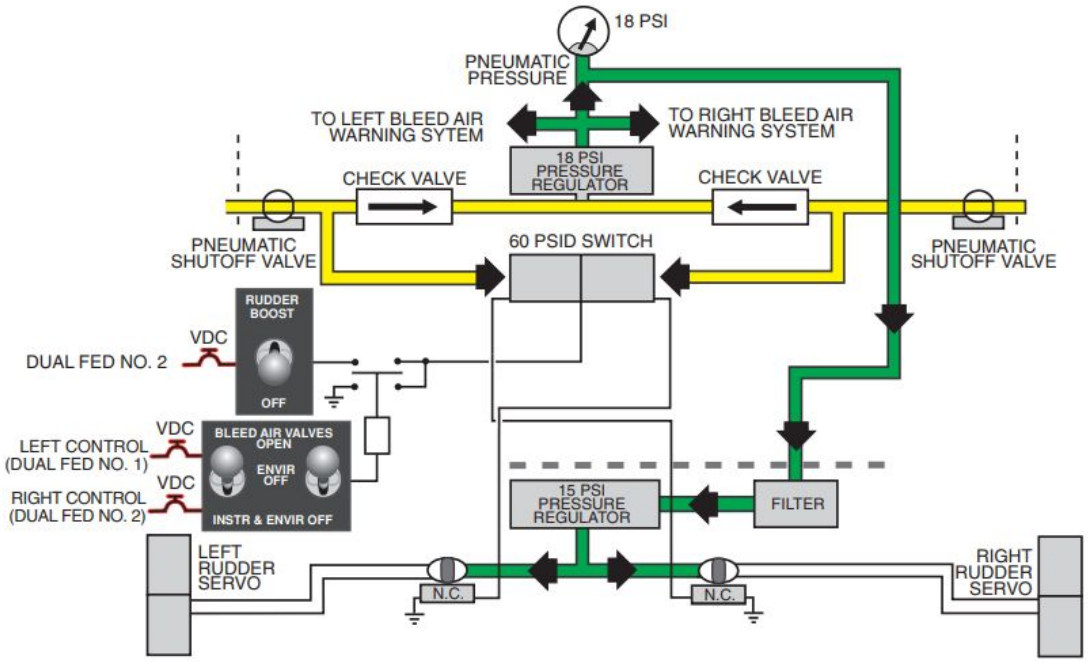

Figura 3.6 Diagrama electroneumático del sistema Rudder Boost. [26]

3.2.3 Chequeo operacional según manual de mantenimiento

Con el interruptor de la fuente de aire estático en la posición **"NORMAL"** y el interruptor del sistema Rudder Boost en la posición **"ON"** encendido este chequeo operacional puede ser realizado durante la corrida de los motores, inicialmente desface el empuje de los motores de tal forma que el empuje diferencial sea tan grande que se active el sistema Rudder Boost,para verificar el correcto funcionamiento se puede inspeccionar el movimiento de los pedales con el cual se opera el timón de dirección cuando el sistema Rudder Boost está activo, si el empuje del motor izquierdo está atrasado, el pedal derecho del timón de dirección debe moverse hacia adelante y si el empuje de motor derecho está atrasado el pedal izquierdo del timón de dirección debería moverse hacia adelante. repita la comprobación de ajustes invertidos para comprobar el movimiento del pedal opuesto del timon de direccion. [26]

NOTA: si el sistema Rudder Boost no opera correctamente realice una prueba funcional como se describe en el **AMM ATA 27-21-00 Página 206.**

3.2.4 Prueba funcional del Sistema Rudder Boost según manual de mantenimiento

NOTA: Todo el seguimiento del sistema del timón de dirección debe completarse antes de la siguiente comprobación. [1]

- Para facilitar la prueba del sistema Rudder Boost como muestra la figura 3.7 a continuación se enumeran los componentes que contiene el equipo de apoyo de tierra análogo del sistema rudder boost:
    - Dos indicadores de presión de una escala de 0 a 100 PSI
    - Dos indicadores de presión de una escala de 0 a 30 PSI
    - Un indicador de presión diferencial 70-0-70 PSI
    - Cuatro válvulas de carga
    - Cuatro válvulas de sangrado
    - Un regulador de presión de una escala de 0 a 15 PSI
    - Un filtro de aire
    - Conectores de tuberías según sea necesario para la conexión del equipo de apoyo en tierra al interruptor de presión diferencial y los servos del sistema Rudder Boost
    - Tubería según sea requerida.
- Compruebe que no existan fugas en los servos del timón de dirección.
    - Remueva los panel de acceso de los lados derecho e izquierdo de la parte posterior del fuselaje y acceda a los servos del timón de dirección.
    - Oprima el interruptor haciendo girar la palanca de sujeción
    - Desconecte las líneas neumáticas de la conexión de los servos y preserve el área, conecte el equipo de apoyo en tierra con el servo del lado izquierdo como se indica en la figura 3,7 , este equipo de apoyo en tierra debe tener la capacidad de enviar una presión de 12 a 15 PSI hacia el servo y realizar un aislamiento de la misma cuando se realizan pruebas , la cual no debe generar fugas.
    - Aplique de 12 a 15 PSI hacia uno de los servos del sistema rudder boost cierra la

válvula de suministro lo cual mantendrá dentro de la línea la presión por 3 minutos lo cual no debe generar fugas al realizar este paso.

- Cierre la línea de presión regulando la presión dentro del sistema desde el control del servo y desconecte el equipo de apoyo en tierra, ahora conecte la línea neumática al servo.

- Repita este mismo procedimiento en el servo del lado derecho.

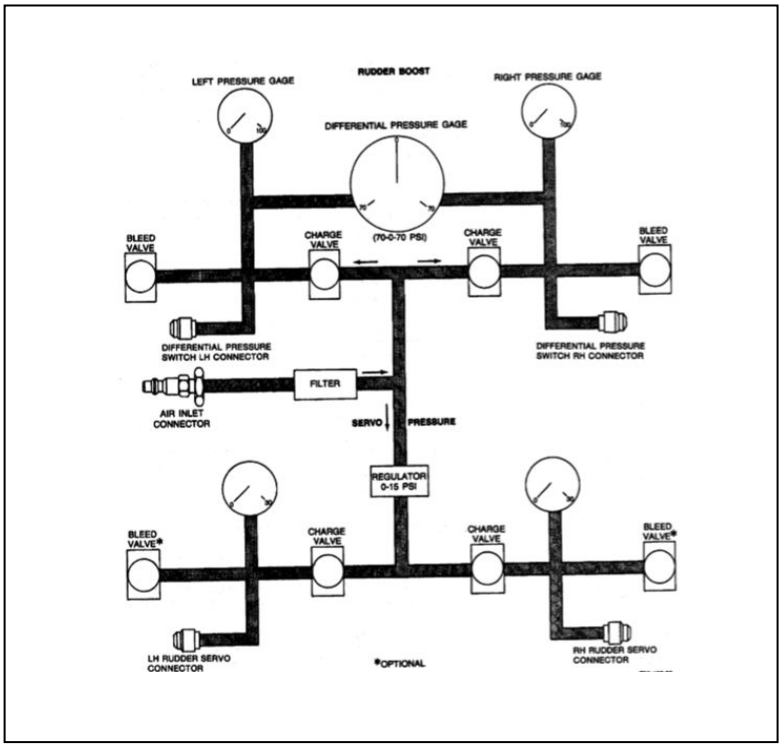

Figura 3.7 Conexión y prueba del equipo apoyo en tierra en el sistema Rudder Boost. [1]

- Compruebe que el sistema de ensamble de resortes elimine la holgura en los cables cuando el sistemas Rudder Boost del timón de dirección esté apagado, compruebe que el sistemas de cables en el timón primario estén libres de movimiento.

  - Conecte la fuente de alimentación externa del avión de acuerdo con las instrucciones descritas en el AMM capítulo ATA 24-40-00.

  - Siga las instrucciones descritas capítulo ATA 07 -00-00 para el levantamiento de la nariz del avión, para que así la rueda del tren de aterrizaje de nariz gire libremente.

  - Desconecte las líneas neumáticas del interruptor de presión diferencial y preservarlas las líneas , no desconecte la conexión eléctrica del interruptor, este interruptor se encuentra debajo de la plataforma del asiento derecho seguido del larguero posterior y se puede acceder a través de un panel de acceso .

  - Conecte el equipo de apoyo en tierra como se indica en la figura 3,7 con los

interruptores de presión diferencial " puntos de conexión del equipo de apoyo en tierra" para simular la entrada de sangrado de aire de motor.

- Conecte un indicador de presión y realice la calibración el cual debe estar dentro de un rango de 0 a 20 PSI al conector en T entra cada valvula alivio y servo de sistema rudder boost, luego realice todas la pruebas en las demás conexiones.

- Desconecte de la línea delantera del sangrado de aire del cortafuegos desde la válvula de cierre de la línea de sangrado de aire del lado izquierdo, conecte una fuente de aire la cual debe contener aire limpio y seco dentro de un rango de 80 a 125 PSI a la válvula de corte.

- Oprima el interruptor del sistema Rudder Boost en la posición encendido que se encuentra en el pedestal , luego oprima el interruptor del sagrado de aire izquierdo en la posición abierta.

- Inspeccione que el timón de dirección esté totalmente deflectado hacia la derecha,

  - Aplique 90 PSI de presión de prueba a ambos lados del interruptor de presión diferencial , minimice la presión del lado izquierdo del interruptor antes de que este actúe, este interruptor deberá operar dentro de un rango de 30 +-4 PSI (presión diferencial de 60 +-4 PSI), la activación del interruptor puede ser verificada por la deflexión del Rudder.

  - El interruptor de presión diferencial no es ajustable si no se obtiene la acción correctiva se debe reemplazar este interruptor.

  - El indicador de presión del servo ubicado en el lado derecho del timón de dirección debe indicar un rango de 15 +-0.50 PSI la presión puede ascender lentamente pero no debe exceder un máximo 16.5 PSI.

  - En cabina presione moderadamente el pedal izquierdo, el timón de dirección se deflectara totalmente hacia el lado izquierdo así se logrará anular la presión que genera el sistema Rudder Boost, luego verifique que la válvula de alivio derecha liberar presión.

  - Descienda la presión del lado izquierdo del interruptor de presión diferencial aproximadamente a 0, luego incremente la presión a 90 PSI y mantengala, el indicador de presión del servo del lado derecho del timón de dirección debe medir que este servo no este recibiendo presión neumática.

- Inspeccione que el timón de dirección esté totalmente deflectado hacia la izquierda

  - Descienda la presión del lado derecho del interruptor de presión diferencial hasta que el interruptor actúa este interruptor deberá operar dentro de un rango de 30 +-4 PSI (presión diferencial de 60 +-4 PSI), la activación del interruptor puede ser verificada por la deflexión del Rudder. El interruptor de presión diferencial no es ajustable si no se obtiene la acción correctiva se debe reemplazar este interruptor.

- El indicador de presión del servo ubicado en el lado izquierdo del timón de dirección debe indicar un rango de 15 +-0.50 PSI la presión puede ascender lentamente pero no debe exceder un máximo 16.5 PSI.

- En cabina presione moderadamente el pedal derecho, el timón de dirección se deflectara totalmente hacia el lado derecho así se logrará anual la presión que genera el sistema Rudder Boost, luego verifique que la válvula de alivio derecha liberar presión.

- Descienda la presión del lado derecho del interruptor de presión diferencial aproximadamente a 0, luego incremente la presión a 90 PSI y mantengala, el indicador de presión del servo del lado izquierdo del timón de dirección debe medir que este servo no este recibiendo presión neumática.[1]

3.3 NEUMÁTICA Y ELECTRONEUMÁTICA

3.3.1 Principios básicos de La Neumática

La neumática es la ciencia que trata las propiedades del aire comprimido, dos de sus principales propiedades son:

- Si se ejerce presión en un gas, esta se transmite con igual intensidad en todas las direcciones.

- Cuando un gas es comprimido dentro de un sistema cerrado, su presión aumenta y también su temperatura.

La neumática es la mecanización que se realiza por medio de las propiedades del aire comprimido en donde se busca la transformación de energía neumática que viene con el gas comprimido a energía mecánica cuando el gas es entregado para generar movimiento en un sistema mecánico.

Un criterio muy importante es la existencia del compresor, si este existe la elección del sistema neumático tiene muchas más posibilidades. Esto es especialmente importante para procesos en equipos de apoyo en tierra aeronáuticos.

3.3.2 Principios básicos y Condiciones de La Electroneumática

La electroneumática hace referencia a la aplicación y combinación de tres importantes ramas, las cuales son la neumática, flujo de aire comprimido y la electrónica. [22]

- Se pueden lograr fuerzas mucho más altas con la presión hidráulica, altas velocidades de operación, menos riesgos de contaminación por fluidos especialmente si se utiliza en la industria de alimentos o farmacéutica, menores costos que la electricidad.
- Alto nivel sonoro, no se pueden manejar grandes fuerzas, el uso del aire comprimido, si no es utilizado correctamente, puede generar ciertos riesgos para el ser humano, altos costos de producción del aire comprimido**.**
- La energía eléctrica sustituye a la energía neumática como el elemento natural para la generación y transmisión de las señales de control que se ubican en los sistemas de mando.
- Los elementos nuevos o diferentes que entran en juego están constituidos básicamente para la manipulación y acondicionamiento de las señales de voltaje y corriente que deberán de ser transmitidas a dispositivos de conversión de energía eléctrica a energía neumática para lograr la activación de los actuadores neumáticos.

3.3.3 Tipos de válvulas neumáticas y electroneumáticas más utilizadas

- **Válvulas neumáticas.** Una válvula neumática es un elemento de regulación y control de la presión y el caudal del aire a presión. Este aire es recibido directamente después de su generación o sino desde un dispositivo de almacenamiento. Las válvulas dirigen, distribuyen o pueden bloquear el paso del aire para accionar los elementos de trabajo. [22]

Figura 3.8 Válvula neumática. [13]

Las válvulas neumáticas tienen una gran importancia dentro del mundo industrial, se clasifican de acuerdo a sus aplicaciones de la siguiente forma: [22]

- **Válvulas de distribución.** Como su propio nombre indica son las encargadas de distribuir el aire comprimido en los diferentes actuadores neumáticos, por ejemplo los cilindros o actuadores, se pueden clasificar de varias maneras, por su construcción interna, por su accionamiento y por el número de vías y posiciones; la clasificación más importante es por el número de vías y posiciones, aunque en este tipo de clasificación no se tiene presente su construcción ni el pilotaje que lleva.
- **Válvulas de bloqueo.** Son válvulas con la capacidad de bloquear el paso del aire comprimido cuando se dan ciertas condiciones en el circuito.
- **Válvulas reguladoras.** Son las válvulas que regulan el caudal y la presión.

Figura 3.9 Válvula Reguladora. [3]

- **Válvulas secuenciales.** Las válvulas neumáticas son considerados elementos

de mando, de hecho, necesitan o consumen poca energía y a cambio, son capaces de gobernar una energía muy superior. Asimismo, cada clase de válvula mencionada tiene sus diferentes tipos:

- **Válvulas de bloqueo.** En este tipo de válvulas tenemos: válvulas anti-retorno, de simultaneidad, de selección de circuito y de escape.

- **Válvulas de regulación.** En esta clase de válvulas encontraremos que tipo de regulación hacen, si son con aire de entrada o de salida, y las válvulas de presión, cuando se habla de la función de la válvula nos estamos refiriendo a la variedad de posiciones de la válvula. Generalmente encontramos de 2/2, 3/2, 4/2, 5/2, 3/3, 4/3 y 5/3".

- **Válvula neumática 2/2.** Esta válvula al igual que la unidireccional es de asiento, es decir que abren y cierran el paso por medio de conos, discos, placas y bolas, evitando cualquier fuga. Estas válvulas son de concepción muy simple, pequeña y económica. Son ideales para gobernar cilindros de simple efecto" disponible en internet.

- **Válvula Anti retorno.** La válvula anti retorno (fig. 3.10) está destinada a impedir una inversión de la circulación del fluido neumático permitiendo dirigir en un solo sentido evitando la formación de mezclas no deseadas por medio de una válvula de retención, minimiza las fugas por medio de una válvula activada por un muelle con estanqueidad vía elastómeros.

Figura 3.10 La válvula anti retorno. [16]

- **Válvulas electroneumáticas.** El dispositivo medular en un circuito electroneumático, es la válvula electroneumática. Esta válvula realiza la conversión de energía eléctrica, proveniente de los relevadores a energía neumática, transmitida a los actuadores o a alguna otra válvula neumática. Esencialmente, consiste en una válvula neumática a la cual se le adhiere una bobina sobre la cual se hace pasar una corriente para generar un campo magnético que, finalmente, generará la conmutación en la corredera interna de la válvula, generando así el cambio de estado de trabajo de la misma, modificando las líneas de servicio. [8]

Figura 3,11 Válvula electroneumática. [24]

- **Válvula liberadora o reguladora de presión (HY- 8200).** También llamadas válvulas de seguridad y su función es liberar un fluido cuando la presión interna de un sistema supera el límite establecido.

  En esta clase de válvulas encontraremos que tipo de regulación hacen, si son con aire de entrada o de salida, y las válvulas de presión,cuando se habla de la función de la válvula nos estamos refiriendo a la variedad de posiciones de la válvula. *[24]*

Figura 3.12 válvula reguladora de presión. [13].

- **Válvula controladora de flujo una sola dirección regulable (HY- 8210):** La válvula controladora de flujo en una sola dirección tiene la función de dejar pasar el fluido como su nombre lo indica en un solo sentido, además se puede regular cuánto caudal queremos que pase a través de la válvula. *[13]*

Figura 3.13 válvula controladora de flujo regulable. [13]

- **Válvula controladora de flujo compensación de presión (HY-8212).** Estas válvulas tienen unos orificios que miden el flujo de aceite y mantienen un flujo constante mientras la presión también se mantiene constante, si la presión cambia, el flujo también cambia automáticamente por lo consiguiente compensa las diferencias de presiones para mantener un flujo constante. Tan pronto como el flujo pasa a través del orificio del control tiende a implementarse su caída de presión y se incrementa proporcionalmente provocando el desplazamiento de la paleta hacia la derecha, este cierre estrangula el orificio provocando una restricción de flujo. [13]

Figura 3.14 válvula controladora de flujo. [33]

- **Válvula reductora de presión (HY- 8220):** Esta válvula tiene la función de mantener una presión constante sin importar las fluctuaciones de flujo o de la presión de entrada. Cuando la presión de la válvula aumenta por encima del ajuste del piloto reductor, el piloto cierra, por lo tanto, el flujo será atrapado en la cámara de control, el pistón se desplazará hacia abajo, la válvula empezará a cerrar, y por ende, la presión disminuirá. La presión disminuirá hasta que esta no sea suficiente para mantener el piloto cerrado. Cuando llegue el momento en que el piloto volverá a abrir, haciendo que la válvula abra nuevamente y que la presión de salida aumente. *[19]*

Figura 3.15 Fotografía válvula reductora de presión. [13]

3.4 COMPONENTES ELECTRÓNICOS

3.4.1 Microcontroladores

Es un dispositivo electrónico que principalmente funciona como un circuito integrado programable con el fin de ejecutar las órdenes grabadas en su memoria, en otras palabras son dispositivos que pueden generar operaciones lógicas.

El microcontrolador se considera también como un sistema cerrado como componente más importante en procesamiento y control que integra en sí mismo todo un conjunto de elementos que lo identifican. [6]

- **Elementos de un microcontrolador:**
    - **Entradas y salidas E/C.** Elemento del microcontrolador que su función es comunicar el circuito integrado con el exterior los pines de conexión pueden ser análogos y digitales existen varios tipos de pines, los pines de entrada en los cuales se puede conectar sensores, y los pines de salida, en los cuales se pueden conectar actuadores.
    - **CPU.** Unidad central de procesamiento también conocida como procesador, es el elemento más importante en un controlador su función es procesar y ejecutar a cabalidad correctamente la programación impartida.
    - **Memoria.** Es el elemento encargado de almacenar toda la información posible tanto los datos como la programación gracias a todo la información guardada la CPU puede tener un gran desempeño y obtener la todos los datos de inmediato.
    - **Memorias persistentes.** Su forma de almacenamiento es permanente

así no se tenga energía eléctrica, el programa que tiene los microcontroladores está contenido en esta memoria en caso de no ser de este modo se perdería de una forma engorrosa la información a falta de la electricidad.

- **Memorias volátiles.** almacenan la información de una manera continua hasta cuando son aisladas de la energía eléctrica. [6]

- **Arquitectura de un microcontrolador:**

La arquitectura de un microcontrolador está basada en su diseño y a nivel de características de rendimiento, las dos principales arquitecturas de microcontroladores que existen son las siguientes:

- **Arquitectura de Von Neumann.** La arquitectura de John Von Neumann se caracteriza porque existe una única memoria en la guardan esta información, la capacidad de almacenamiento está fijada por el ancho de bus de datos, este espesor es la cantidad de información que se puede transferir por unidad de tiempo.

El siguiente esquema representa este tipo de arquitectura:

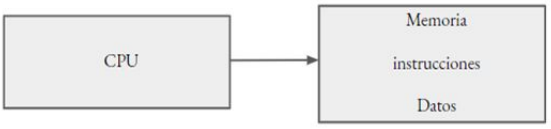

Figura 3.16 Esquemático arquitectura de John Von Neumann. [6]

Esta Arquitectura presenta las siguientes discrepancias:

- La longitud de las instrucciones impartidas está limitada por el tamaño de los datos es decir para lenguajes algorítmicos complejos es indispensables el alcance de varias memorias.

- Por poseer un solo bus de datos respecto a su velocidad de transferencia de operación de datos se puede ver afectada.

- **Arquitectura de Harvard.** La Arquitectura de Harvard divide la memoria conjunta de instrucciones y datos que vemos en la arquitectura de Von Neumann en dos memorias distintas, una en la que almacena datos y en la otra almacena datos. *[6]*

El siguiente esquema representa este tipo de arquitectura:

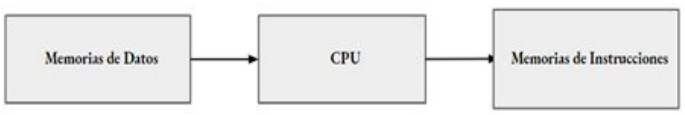

Figura 3.17 Esquemático arquitectura de Harvard. [6]

La memoria de instrucciones es llamada memoria de programa y únicamente almacena las instrucciones de programa que ejecuta la CPU la memoria de datos almacena únicamente los datos que utiliza la CPU y es llamada memoria de datos.

La mayor ventaja que tiene la arquitectura de harvard respecto a la arquitectura de Von Neumann es que al contar con dos memorias y se tiene dos buses de datos independientes y con posibilidad de tener diferentes anchos de bandas, lo cual permite que la arquitectura Harvard elimina las discrepancias encontradas en la arquitectura de Von Neumann.

La tendencia actual de la arquitectura de los microcontroladores es utilizar la arquitectura Harvard.

3.4.2 Procesadores

Las dos arquitecturas mencionadas anteriormente pueden utilizar los diferentes tipos de procesadores que existen.

- **Procesadores CISC.** Los procesadores CISC conocidos como complex instruction set computer,permiten manejar un conjunto complejo de instrucciones para generar una programación altamente completa.
- **Procesadores RISC.** Los procesadores RISC conocidos como (reduced Instruction Set Computer) utilizan una menor cantidad de instrucciones, además de esto se caracterizan por enlazar instrucciones que tienen que ejecutar en lapso de tiempo más corto caracterizados por ser los procesadores más rápidos*[6]* .

3.4.3 Tarjetas microcontroladoras

Es una composición de varios elementos electrónicos contenidos en una placa de material semiconductor que interactúan entre sí para procesar señales electrónicas, esta placa tiene la capacidad de contener varias conexiones de entradas y salidas pueden conectarse diversos dispositivos para procesar esta información que va a ser destinada al sistema que se está controlando, es llamada tarjeta controladora porque por medio de su programación puede contener varios procedimientos lógicos que dan instrucciones a componentes electromecánicos y así controlar por ejemplo equipos que cumplan con aplicabilidad electroneumática como lo son los equipos de apoyo en tierra que funcionan con principios

neumaticos.

- **Tarjeta controladora Arduino Mega 2560:**

Es una tarjeta que porta 54 pines los cuales pueden configurarse para establecer salida o entrada de señal, 14 pines se usan con la función de modulación por ancho de pulso PWM, tiene 16 entradas analogicas y 4 puertos para la comunicación serial.

La velocidad del microcontrolador contiene un cristal de 16MHz y memoria flash de 256k. Maneja un rango de voltaje de 7 a 12 Vdc. La comunicación entre la tarjeta y la computadora está dada por el puerto serie por el cual habrá transmisión y recepción del computador a la tarjeta y viceversa.

Figura 3.18. Tarjeta Arduino Mega 2560. [30]

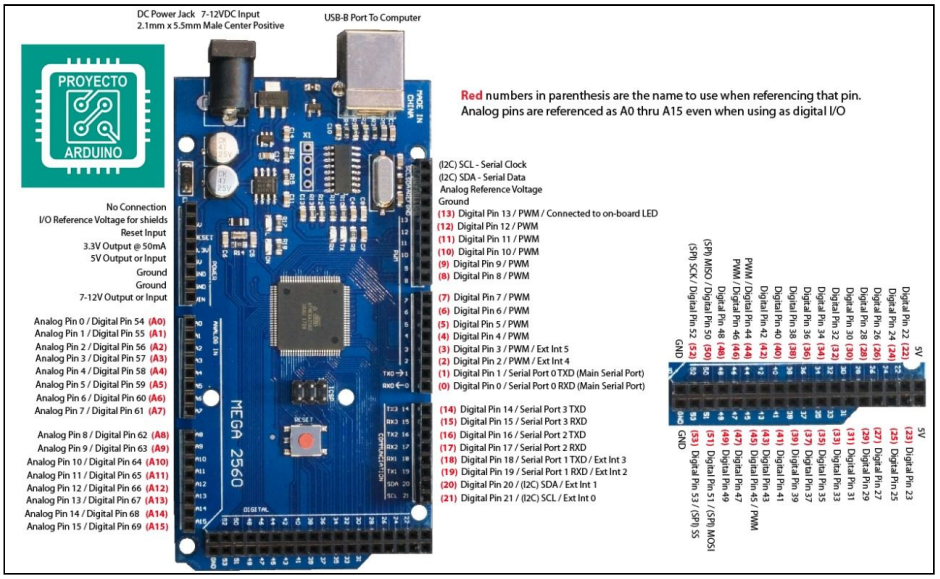

Figura 3.19 Indicativos de las señales de entrada y salida de los pines que tiene la tarjeta

controladora Arduino Mega. [34]

- **Pines de alimentación.** La placa de arduino incorpora un conjunto de pines que están relacionados con la alimentación de los componentes electrónicos que forman los circuitos. Puede estar localizado en la sección de pines indicados como Power. los pines relacionados con la alimentación eléctrica son los siguientes

  - **3,3v:** pin hembra que ofrece 3.3 voltios y que está regulado mediante un circuito integrado es utilizado para alimentar aquellos componentes electrónicos que necesiten esta cantidad de voltaje, y es obtenido mediante las diferentes fuentes de alimentación que pueden tener la placa

  - **5v:** pin hembra que ofrece 5 voltios y puede alimentar la propia placa desde la fuente de alimentación externa previamente regulada y componentes que requieren 5 voltios para funcionar correctamente.*[14]*

  - **GND:** En todo circuito electrónico los componentes necesitan estar conectados a tierra en en este pin hembra ofrece una tierra común para todos los componentes.

  - **Vin:** Pin hembra que puede ser utilizado para alimentar componentes electrónicos del nivel de voltaje que recibe la placa sin regular y la alimentación de la fuente externa. Vin se conecta el borne positivo a la fuente y en el GND el borne negativo, la tensión suministrada por la fuente externa es regulada por el regulador de tensión que incorpora la placa.

- **Pines con doble función Arduino:** La placa de arduino incorpora una serie de pines dentro de los pines de entrada y salida digitales y de entrada analogicas que, ademas de su función propia de entrada y salida, incorporan una funcionalidad extra a la placa.

  - **Pin 0 (RX):** pin que permite al microcontrolador ATmega 328P aplicable a la placa arduino arduino 1 Revisión 3 recibir directamente datos en serie sin pasar por el conversor USB serie.

  - **Pin 1 (TX):** pin que permite a la placa arduino arduino 1 Revisión 3 enviar directamente los datos de serie sin pasar por el conversor USB serie.

  - **Pin 2:** Mediante programación software se puede utilizar para gestionar interrupciones en la placa.

  - **Pin 3:** Mediante programación software se puede utilizar para gestionar interrupciones.

  - **Pines para comunicaciones serie con el protocolo SPI:** usado principalmente para la transferencia de información entre circuitos integrados en equipos electrónicos.

- **SPI:** es otro protocolo serie muy simple. Un maestro envía la señal de reloj, y tras cada pulso de reloj envía un bit al esclavo y recibe un bit de éste. Los nombres de las señales son por tanto SCK para el reloj, MOSI para el Maestro Out Esclavo In, y MISO para Maestro In Esclavo Out. Para controlar más de un esclavo es preciso utilizar SS (selección de esclavo).

- **Pin 10: SS**

- **Pin 11: MOSI**

- **Pin 12: MISO**

- **Pin 13: SCK**

- **Pin 13:** Permite detectar de forma rápida y sencilla señales externas sin necesidad de conectar componentes extras a nuestro circuito.Esto se debe a que está conectado a un led incrustado en la placa que se encenderá si la señal recibida es high y se apagará si la señal es low.*[14]*

○ Salidas analogicas Arduino:

La placa arduino arduino mega dispone de 16 salidas analogicas dedicadas exclusivamente a esa función, La razón de que la placa arduino disponga de dichos pines es porque es capaz de manejar este tipos de señales y lo que hace es simular mediante pines de entrada y salidas digitales.

Este tipos de salidas analogicas simuladas mediante entradas y salidas digitales se llaman salidas analogicas PWM, pulse width Modulation y su modo de funcionamiento es enviar señales formadas por pulsos de frecuencia constante en lugar de una señal continua, es decir, variando la frecuencia y anchura de los pulsos que se envían , simulando la tensión que será de forma continua.

Existe una limitación en las placas y no solo en las de arduino, que permiten proporcionar salidad analogicas autenticas y para ello , utilizan un mecanismo de simulación de salida analogica para lograr una comunicacion con los actuadores que estan en la capacidad de recibir entradas analogicas. *[14]*

○ Entradas y salidas digitales Arduino:

La placa de arduino dispone de 14 entradas y salidas digitales que se encuentran en la sección de pines indicados como el entorno digital se encuentran enumerados del 0 al 13 comúnmente son conocidos como GPIO “General Purpose Input/Output.

La función principal de las entradas y salidas digitales es la conexión de los sensores y actuadores para recibir y enviar información desde y hacia los componentes, todos funcionan a 5 voltios y disponen de una resistencia interna, pull up, que debe activar mediante código fuente si se necesita utilizarla.

Los pines al momento de generar intensidad a los componentes conectados a ellos, se

encuentran agrupados en dos grupos , pines de 0 a 4 y pines de 5 al 13.

Cada grupo de pines genera un máximo de 100 mA de tensión a la vez, siendo 40mA el máximo que cada pin puede proporcionar .

- **Entradas Analógicas Arduino:**

La placa arduino dispone de 6 entradas analogicas que se encuentran en la sección de pines indicados como **ANALOG IN.** Los pines se encuentran numerados empezando por A0 y acabando en el A5 y puede recibir valores de voltajes en un rango de 0 a 5 voltios.

Esta placa arduino solo trabaja con valores digitales por los que los valores hallados en cada uno de los pines son transformados a valore digitales aproximados por el circuito conversor analogico/digital que incorpora la placa.El nivel de precisión de dicha conversión viene dado por el número de bits que incorpora el circuito de conversión de una forma exponencial , a mayor número de bits mayor será la precisión.

Los pines pueden ser utilizados como pines de entrada y salidas digitales en caso de necesitarlo por la complejidad del circuito, en este caso cambia la numeración y el lugar de utilizar la posición A0 a A5, estarán numerados del 14 al 19.

- **Alimentación de las placas arduino:**

El tipo de alimentación para las placas puede obtenerse de 3 formas diferentes:

- Conexión de la placa arduino a la fuente externa.
- Conexión de la placa arduino mediante USB a un ordenador.
- Conexión a un puerto de alimentación.

La placa de arduino necesita 5 voltios de alimentación para lograr funcionar correctamente en el caso de las conexiones externas, la placa está preparada para recibir entre 6 y 20 voltios, mediante un circuito regulador de tensión que tiene incorporado la placa y que convierte el voltaje de entrada en 5 voltios continuos.

En el caso de la conexión mediante USB el voltaje recibido siempre es de 5 voltios, y el caso de recibir una tensión mayor en el puerto USB, la placa arduino está capacitada para detener la conexión mediante el polifusible reseteable que incorpora hasta que se restauren los 5 voltios.[14]

En el caso de una conexión a un puerto de alimentación , el voltaje que recibe dicho puerto tiene que estar previamente regulado a 5 voltios.

- **Puerto USB Arduino:**

El puerto USB de la placa arduino es el principal puerto mediante el cual se genera enlace, con los demás dispositivos que se encuentran al exterior, sus funciones principales

son:

- Alimentación de la placa.
- Cargar los programas en el microcontrolador desde el ordenador.
- Envío de la información desde la placa al ordenador y viceversa.

○ Pines para comunicaciones serie con el protocolo 12C Arduino Mega:

Es el que utiliza dos líneas para transmitir la información: una para los datos y otra para la señal de reloj. También es necesaria una tercera línea, pero esta sólo es la referencia (masa). Como suelen comunicarse circuitos en una misma placa que comparten una misma masa esta tercera línea no suele ser necesaria.

Las líneas **SDA** y **SCL** son del tipo drenaje abierto, es decir, un estado similar al de colector abierto, pero asociadas a un transistor de efecto de campo (o FET). Se deben polarizar en estado alto (conectando a la alimentación por medio de resistores "pull-up") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas.

- **Pin A4: SDA**
- **Pin A5: SCL**
- **Reset:** Permite reiniciar la placa al establecer su valor a Low
- **AREF:** Permite aumentar la precision de las entradas analogicas ofreciendo un voltaje de referencia externo
- **Pin sin uso:** reservado para futuros usos. [14]

• Tarjeta controladora Launchpad Texas Instrument MSP430G2553:

La tarjeta Texas instrument **MSP430** contiene un microcontrolador **MSP430G2553** (MCU) de Texas Instruments (TI) son Procesadores de señal mixta basados en la arquitectura RISC (**reduced instruction set computer o computador de conjunto de instrucciones reducidas**) de 16 bits diseñados específicamente para Ultra bajo consumo de energía. Esta tarjeta controladora tiene la combinación correcta de periféricos inteligentes, facilidad de uso, bajo costo y menor consumo de energía para miles de aplicaciones. [14]

○ Componentes de la tarjeta Texas Instrument MSP430:

Cómo componente principal se tiene el **Microcontrolador MSP430G2553** que es un microcontrolador de señal mixta de ultra baja potencia con 16 incorporados temporizadores de bits, hasta 24 pines capacitivos táctiles de E /S habilitados, un comparador

analógico versátil y comunicación incorporada y una Capacidad usando la interfaz de comunicación serial universal. Además, los miembros de la familia MSP430G2553 tienen un convertidor analógico a digital de 10 bits (A / D).

El dispositivo cuenta con un potente CPU RISC de 16 bits, registros de 16 bits y generadores constantes que contribuyen a la máxima eficiencia del código, el oscilador controlado digitalmente (DCO) permite la activación de los modos de baja potencia al modo activo en menos de 1 microsegundo (μs).

- **Características principales del microcontrolador MSP40G2553:**

  - Velocidad del reloj: Configurable entre 1 y 16 MHz
  - Memoria FLASH: 16 KB
  - Memoria SRAM: 512 KB
  - Memoria NVM: 56 KB
  - Memoria SRAM: 4 KB
  - Pines GPIO: 24 como máximo.
  - 2 Temporizadores
  - Convertidor ADC de 8 canales
  - UART
  - 12C
  - SI

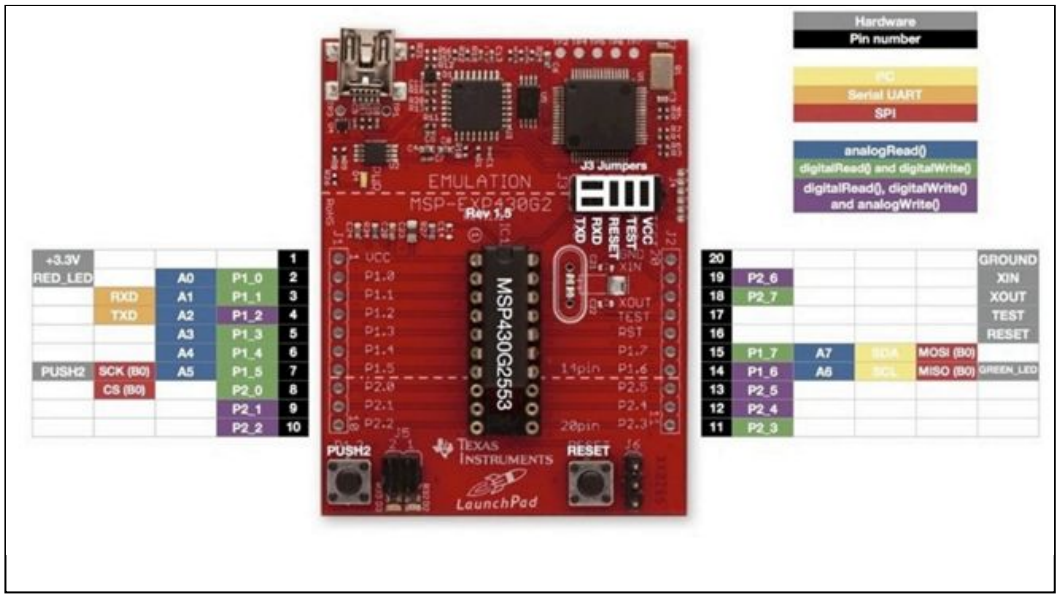

Figura 3.20 Placa de desarrollo launchpad y las respectivas funcionalidades de su microcontrolador. [23]

La plataforma Launchpad ha sido la respuesta de Texas Instruments a la necesidad de tener un sistema de bajo costo, amigable y que continúe con la filosofía iniciada por Arduino[23].

El primer elemento de la familia en aparecer fue el Launchpad de la familia

MSP430G2, al que han seguido multitud de ellos, con microcontroladores cada vez más potentes y mayor cantidad de periféricos.

El sistema se alimenta a través del puerto USB, aunque también se puede alimentar externamente con una fuente de no más de 3.6V, siendo ésta su máxima tensión.

El microcontrolador es de bajo consumo llegando a consumir por debajo de 1μA en modo de reposo. La placa es suministrada con dos microcontroladores distintos, de la misma familia del MSP43G2553. [28]

- **Características principales tarjeta Launchpad Texas instrument:**
  - Bajo precio, en relación a otras plataformas
  - Posibilidad de programar en bajo nivel (lenguaje C) o con un entorno idéntico al de Arduino (llamado Energía)
  - Microcontrolador en formato DIP (dual in line package) paquete dual en línea, que puede ser extraído de la placa y ensamblado en prototipos independientes.
  - Gran cantidad de periféricos, incluyendo sensores capacitivos en sus pines. [28]

La plataforma Launchpad de Texas Instruments, centrada en el Launchpad MSP430 programada con el entorno de desarrollo suministrado por el fabricante,Por sus características técnicas, este sistema no ofrece mucha capacidad para manejar sistemas de control complejos, pero resulta muy sencillo de programar y tiene una estructura interna muy accesible.

Es justo reconocer que no es posible hacer la comparación de manera totalmente objetiva, por lo que la utilización de las dos genera una total integración y como complemento de manera general, se observa que las plataformas de Texas Instruments son las más económicas y en general, dentro de los grandes fabricantes de semiconductores. Por otro lado, no cabe duda que Arduino, al ser la primera en conseguir establecerse como estándar en el segmento académico, dispone de la mayor base de programas y desarrollos en la web.[28].

- **Entradas y salidas Analogicas de la placa de desarrollo launchpad:**

La placa de desarrollo launchpad cuenta con Entradas analógicas: 8 entradas multiplexadas a un solo ADC – ADC de 10 bits, 200ksps – Rango de entrada (VREF) seleccionable: l VCC l 2.5V l 1.5V l Tensión externa, además cuenta con periféricos incluyendo entre otros un convertidor A/D de 10 bits y 500 ksps, comparador analógico, 2 timers de 16 bits con 3 comparadores cada uno, 2 puertos serie configurables como spi, i2c o

uart, 16 pines de e/s.

- **Entradas y salidas digitales de placa de desarrollo launchpad:**

La plataforma de desarrollo launchpad cuenta con Puerto SPI , Modo Master y Slave –3 y 4 hilos (con CS) – Hasta 10MHz – Usos típicos: conexión de pantallas, memorias... l Bus I2C: – Modo Master y Slave – Hasta 400 Kbps – Usos típicos: sensores con salida I2C, memorias I2C l Puerto UART: – Disponible en el conector y a través del USB – En el USB, limitado a 9600bps – Usos típicos: monitores de depuración, comunicación con ordenadores

Además cuenta con dos temporizadores (A y B) – 16 bits – 4 modos de operación (stop, continous, up, up/down) ,2 ó 3 registros de comparación o captura – Salidas configurables con capacidad PWM(modulación por ancho de pulso , Interrupciones l Reloj: – 3 fuentes: LFXT1, DCOCLK, VLOCLK – 3 señales de reloj para elegir: ACLK, MCLK, SMCL

- **Salidas analógicas placa de desarrollo launchpad:**

Cuenta con una CPU Risc de 16 bits a 16 MHz 16K de memoria Flash para programas y 512 bytes de ram. Gran riqueza de periféricos incluyendo entre otros un convertidor A/D de 10 bits y 500 ksps, comparador analógico, 2 timers de 16 bits con 3 comparadores cada uno, 2 puertos serie configurables como spi, i2c o uart, 16 pines de e/s.

- **Pines con doble función placa de desarrollo launchpad:**

Hay que tener en cuenta que estos pines I/O, cuando funcionan como entradas, necesitan de resistencia de *Pull Up* para poder detectar las señales de los pulsadores o botones, ya que estos funcionan a nivel bajo (conectándose a tierra). Para colocar una resistencia de *pull up* a estos pines se puede hacer, bien por hardware, añadiendo unas resistencias a la placa, entre Vcc y la línea que conectaría el botón al pin, o bien por software, utilizando los registros P1REN o P2REN y luego P1OUT o P2OUT (en estos últimos escribiendo un 1 o a 0 en el pin deseado para indicar si queremos en dicho pin resistencias de *Pull Up* o de *Pull Down,* respectivamente). Al hacerlo así, el propio MSP430 añade las resistencias de *pull up* o *pull down* internamente, simplificándonos el circuito impreso a diseñar. *[23]*

3.5 PRINCIPIO DE LÓGICA Y PROGRAMACIÓN

El hecho de crear un programa implica recurrir a la escritura y aprender un lenguaje. Similar a cuando aprendemos un nuevo lenguaje oral o escrito, necesitamos aprender una sintaxis y una lógica. Escribir en un lenguaje humano es muy complicado. Posee ambigüedad en las palabras, y mucha flexibilidad para la construcción de párrafos. En ese sentido, el lenguaje de máquina es más sencillo, pero posee una dificultad clara: la lógica de la

programación. Usualmente, un ser humano puede percatarse de un error de sintaxis y darlo por alto, restándole parcial o completa importancia. Sin embargo, en un lenguaje de máquina las cosas son de una forma o de otra: O está bien, o está mal. [28]

3.6 SOFTWARES DE PROGRAMACIÓN Y SIMULACIÓN

3.6.1 Software Arduino

Es una plataforma de edición y configuración a partir de un código de programación que posee múltiples herramientas de desarrollo para operación de hardwares que tienen cómo función la ejecución de varias instrucciones indicadas por el código fuente, el conjunto de placas de desarrollo Arduino se programan a través de este software, es el único compatible dado por el mismo fabricante para su uso en conjunto, a continuación en la *Figura 3.21 Entorno de programación software arduino. [34]* se muestra la plataforma inicial del software arduino.

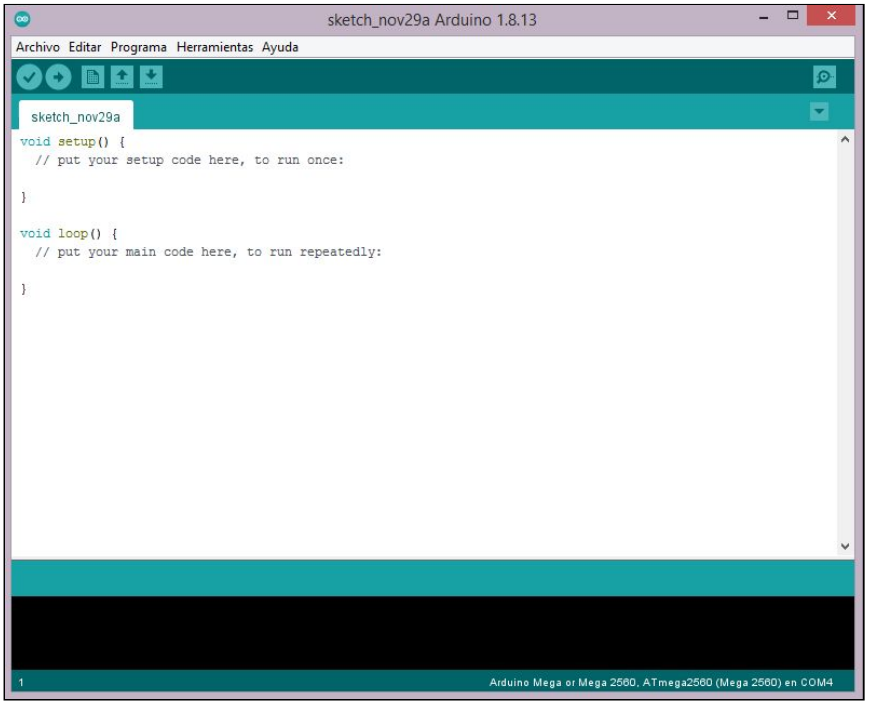

Figura 3.21 Entorno de programación software arduino. [34]

Este entorno de programación contiene un menú de opciones, en algunas de estas se encuentran ejemplos guía para analizar de qué forma se puede editar el código de

programación, configurar la comunicación entre la tarjeta y el software, seleccionar la tarjeta que se va a programar para la compatibilidad, recibir ayuda, sugerencia y demás funciones necesarias para la programación.

El software establece que el código debe comprender varias condiciones lógicas, cómo lo son el nombramiento de variables, la activación de los pines y la descripción de instrucciones, esto está estandarizado en el lenguaje de programación por medio de una sintaxis establecida para cada instrucción.

3.5.3 Software Energía

El software ENERGÍA es un entorno de desarrollo principalmente consta de un editor para la escritura del código, una consola, una línea de mensajes, barra de herramientas para las funciones comunes igual que los menús. se conecta al hardware LaunchPad con el propósito de crear aplicaciones y controlarlas.

Está plataforma tiene el mismo principio de funcionamiento, comunicación y sintaxis a comparación del software arduino, son libres y de uso general para muchos fines de programación electrónica académica e industrial, en la *Figura 3.22 Entorno de programación software Energía. [28]* el entorno de programación principal de Energía.

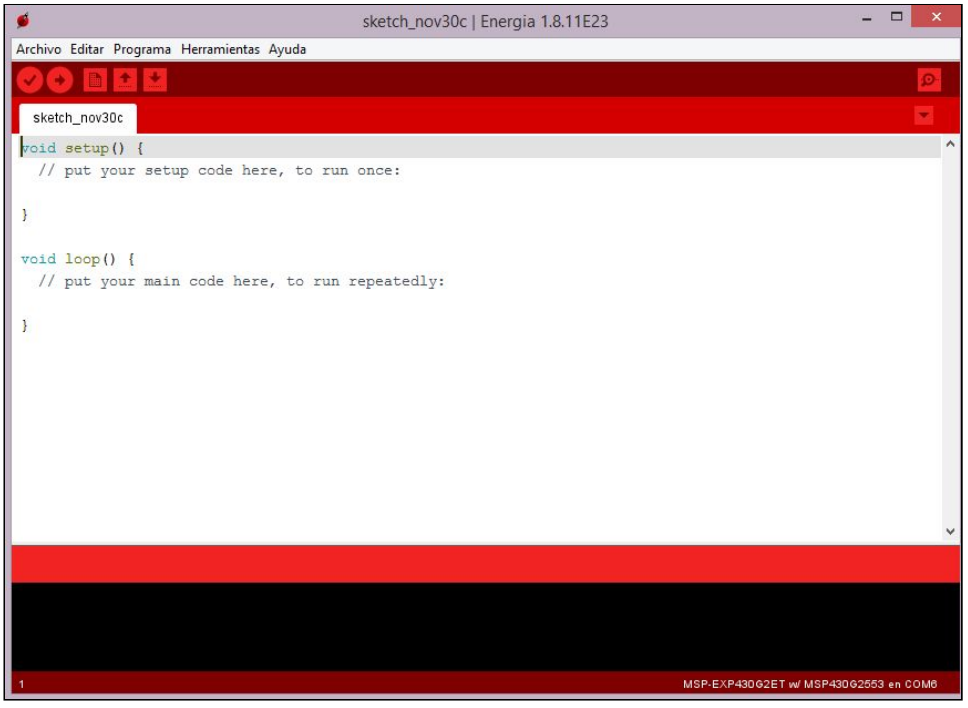

Figura 3.22 Entorno de programación software Energía. [28]

3.5.4 Software Processing

Dado el avance de los medios de producción multimedial, nace un potente software dedicado a la producción de imágenes, animaciones e interactivos. El software denominado Processing. El proyecto da inicio en el 2001, realizado por Casey Reas y Ben Fry, a partir de terminologías realizadas en el MIT Lab, dirigido por John Maeda, e influenciado por el proyecto Design by Numbers. Es así que Processing se convierte en un poderoso entorno de producción basado en Java. [31]

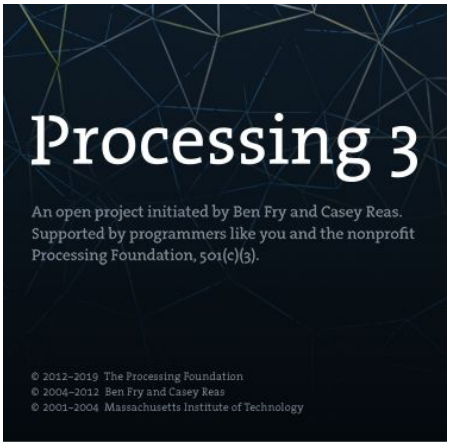

Figura 3.23 software processing [31]

Processing es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, que sirve como medio para la enseñanza y producción de proyectos multimedia e interactivos de diseño digital. Es desarrollado por artistas y diseñadores como una herramienta alternativa al software propietario. Puede ser utilizado tanto para aplicaciones locales como para aplicaciones en la web. [31]

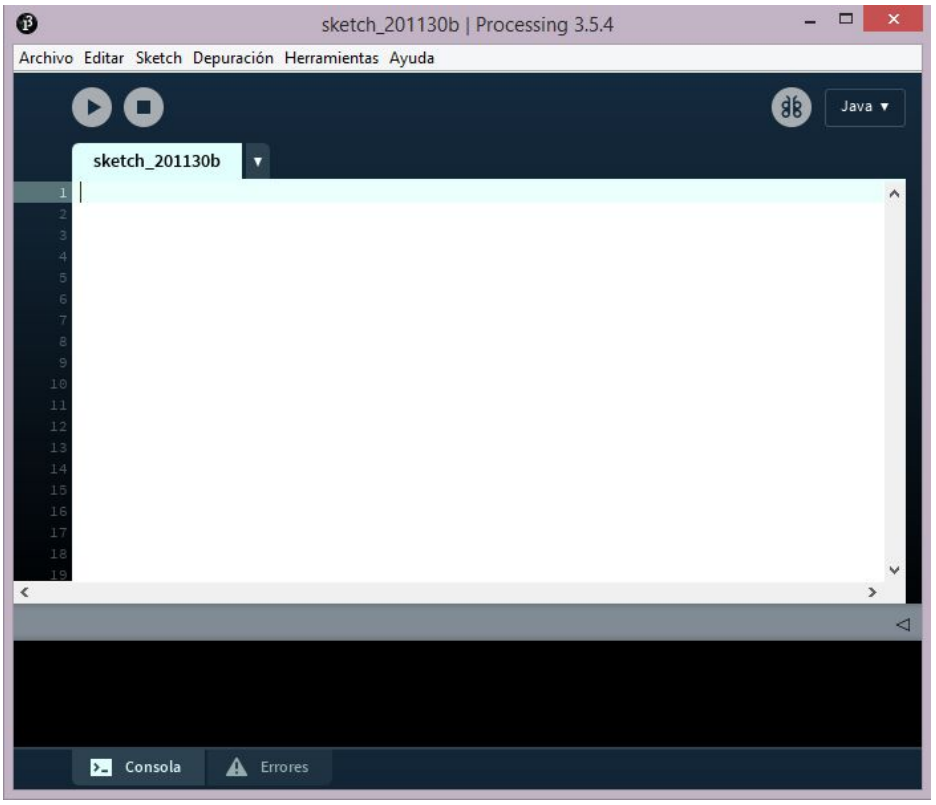

Figura 3.24 Entorno de programación software processing [31]

- **Ambiente de programación de Processing Fundamentos:**
  - **Elementos principales:**
    - Tipos de datos - Variables
    - Instrucciones repetitivas - for() loop
    - Propiedades de la forma - fill - stroke - background
    - Movimiento Métodos - setup() - draw()
    - Instrucciones condicionales - if ()
    - Respuesta y Estímulo - Input (mouse)
    - Dibujo Herramientas de dibujo - Estructuras:
  - **Funciones:**

Una función puede definirse como un conjunto de instrucciones que permiten procesar las variables para obtener un resultado. Este modo llamado continuo se basa en dos construcciones: setup y draw.

Las funciones permiten dibujar formas, colores, realizar cálculos

matemáticos, entre otras variadas acciones. Por lo general se escriben en minúsculas y seguidas por paréntesis. Algunas funciones aceptan parámetros, los cuales se escriben entre los paréntesis. Si acepta más de uno, son separados por una coma (,). A continuación un programa que incluye dos funciones: size() y background(). //Con esta función establecemos el tamaño de la ventana de presentación //El primer parámetro corresponde al ancho de la ventana //El segundo parámetro corresponde al alto. size(400, 300); //Con esta función establecemos el color de fondo de la ventana //Acepta diversos parámetros, para una escala de grises bastará con valores de 0 (negro) a 255 (blanco). background(0); *[31]*

- **Conceptos:**
  - **Instrucciones:** Son los elementos estructurales del programa. Todas las instrucciones deben finalizar con ";" Ejemplo: point (100,100);
  - **Comentarios**: Son usados para hacer notas (apuntes) entre las líneas de código que por lo general facilitan la comprensión del programa. Para hacer comentarios de una línea se debe iniciar el comentario con "// "
  - **Comentarios que requieren más de una línea**: Se debe iniciar el comentario con "/*" y finalizarlo con "*/ "
  - **La función setup:** se ejecuta una única vez cuando se inicializa el programa. Se utiliza para definir las propiedades iniciales del ambiente como el color de fondo, cargar imágenes, inicializar variables. Sólo puede existir una función setup por sketch.
  - **La función draw:** se ejecuta continuamente y es utilizada para dibujar elementos en pantalla. El número de veces que la función draw es ejecutada puede ser controlado por la función delay o suministrando el número de marcos por segundo con la función framerate. También es posible ejecutarla una sola vez incluyendo la llamada a la función noLoop en setup [31]

Capítulo 4

METODOLOGÍA

La investigación para el desarrollo de este proyecto es de tipo exploratoria y correlacional, con base en los objetivos propuestos, se establece inicialmente simular mediante los softwares de programación Arduino y Energía, un equipo de apoyo en tierra operado de forma sencilla y manualmente por medio de un panel de operación, para verificar el óptimo chequeo funcional y pruebas satisfactorias en el sistema rudder boost del King Air Series 200.

Por otra parte se propone una programación de un código que unifique, relacione y use una sintaxis de condiciones lógicas ya estudiadas de los softwares que cumplan el objetivo de controlar el equipo de apoyo en tierra propuesto por medio de una tarjeta microcontroladora, además el estudio exploratorio abarca la obtención de conocimiento sobre los principios físicos de los componentes que conforman el equipo de apoyo en tierra para luego implementar el estudio correlacional que en este caso se refiere al conocimiento necesario para plantear el comportamiento de los componentes ensamblados y cómo a partir de la tarjeta y el código algorítmico, estos pueden ser controlados.

A continuación se muestra un diagrama de flujo en donde se ilustra el procedimiento general que se debe llevar a cabo para desarrollar adecuadamente la respectiva simulación cumpliendo con los objetivos propuestos.

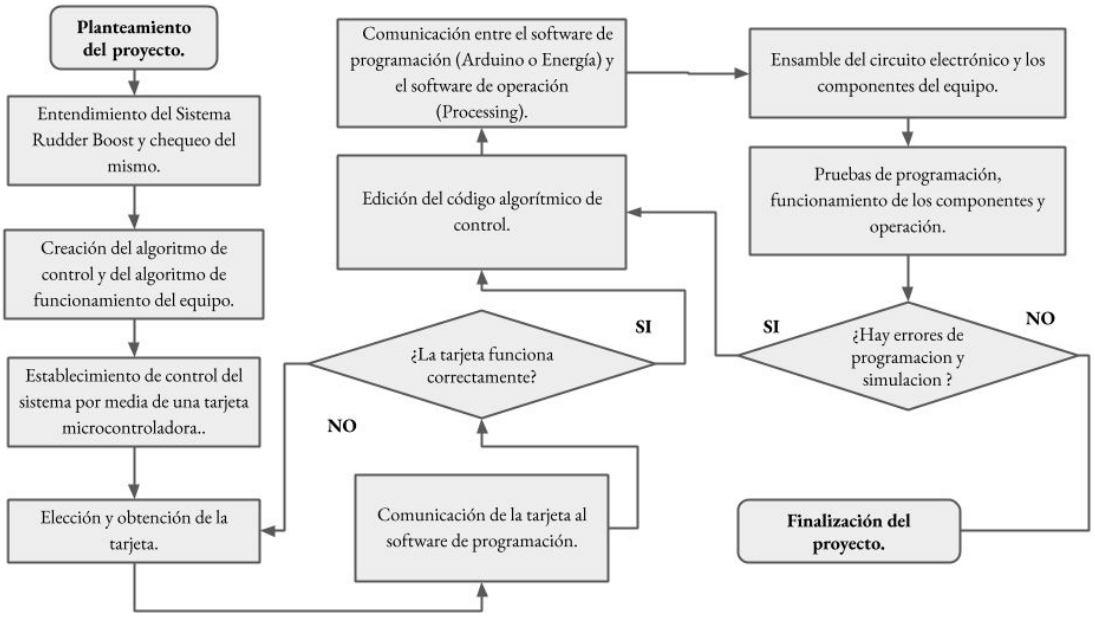

Figura 3.25 Procedimiento general para el desarrollo de la simulación. [6]

Capítulo 5

ANÁLISIS DE RESULTADOS

5.1 PLANTEAMIENTO DEL CIRCUITO ELECTRÓNICO DEL EQUIPO

5.1.1 Análisis General

A Partir de los requerimientos dados en el manual de mantenimiento para el desarrollo de la prueba y chequeo del sistema Rudder Boost de la aeronave, se evidencia que las presiones usadas para esta prueba son estandarizadas y se encuentran en el rango de presiones usadas industrialmente no sólo para pruebas de este sistema y está aeronave sino para otros sistemas en general en la misma aeronave u otro modelo de aeronave que tenga los mismos parámetros de de operación.

Para el desarrollo del diseño se empieza a partir del proceso instructivo descrito para esta prueba, con base en este, se debe llevar a cabo un algoritmo de control cómo inició fundamental del diseño del equipo, este algoritmo de control y operación es la proposición de las instrucciones que se ejecutarán una por una con el fin de cumplir el objetivo de la prueba, el análisis de cada instrucción es la tarea del ingeniero para deducir qué componentes deben usarse para ejecutar cierta instrucción de prueba de acuerdo al diseño, ahora toma lugar en el análisis a establecer de cómo los componentes seleccionados van a interactuar entre sí para llevar a cabo el cumplimiento de cada proceso con éxito.

Para los equipos automatizados la herramienta fundamental de control son las tarjetas controladoras que tiene cómo función enviar y recibir información a un sistema según la configuración previamente asignada a la misma, por medio de un código de programación establecido a la tarjeta mediante un software compatible, este código tiene unas condiciones de lenguaje, es decir que las instrucciones descritas en el código estandarizan el procesamiento de información que envía y recibe la tarjeta para controlar los componentes de forma sincronizada, este control es el que determina que los componentes cumplan una función algorítmica que tendrán cómo fin el desarrollo de cada instrucción.

5.1.2 Análisis de funcionamiento del equipo:

- **Principio de automatización del sistema electroneumático del equipo**

El equipo a diseñar tiene cómo finalidad realizar una prueba de chequeo con principios de automatización electroneumática de la forma más sencilla posible, es decir que los procesos dados por el manual sean reducidos siempre y cuando el equipo este capacitado para omitirlos, por ejemplo, la calibración inicial de un equipo realizada manualmente antes de su uso, es un proceso que no está automatizado y requiere de un procedimiento más del operador para el ajuste de la presión, ya que esta presión de entrada tiene un porcentaje de error a la presión de salida deseada, que es la que se va a usar para las tareas asignadas; en este caso el equipo a diseñar calibra la presión dentro del mismo sistema por medio del control programado para que la presión de salida sea siempre la deseada, teniendo en cuenta que la presión de entrada esté dentro de un intervalo de presión aceptable.

- **Descripción de las funciones que debe cumplir el equipo:**

En primer lugar para el diseño es importante describir las funciones que el equipo debe cumplir en la prueba de chequeo del sistema Rudder Boost, las cuales son en términos generales, el suministro de presión en un rango de 90 PSI a 125 PSI por dos líneas, estas presiones se deben controlar según lo requiera en la prueba de chequeo, de forma tal que mientras una línea está suministrando presión la otra línea no y viceversa o en un tercer caso las dos líneas suministren la misma presión al mismo tiempo.

- **Componentes electrónicos requeridos para el óptimo funcionamiento del equipo:**

Los componentes que aplican para la conformación del sistema electroneumático del equipo y el desarrollo de su operación son dos válvulas solenoides electroneumáticas normalmente cerradas controladas por una única señal de tensión para abrir y cerrar mediante esta señal; una válvula proporcional electroneumática controlada con una señal de corriente y alimentada con una señal de tensión estable para la conversión de presión de entrada a la presión de salida deseada; una válvula solenoide electroneumática normalmente abierta controlada con una señal de tensión para abrir y cerrar que tiene como función liberar la presión del sistema en caso de ser necesario; el mando de control estará dado por una tarjeta microcontroladora la cual se encarga de enviar y recibir las señales de variaciones de tensión y corriente a los componentes para activarlos o desactivarlos de forma sucesiva siguiendo el paso a paso del procedimiento estandarizado; la información de magnitud de presión inicial será tomada por un transductor de presión (sensor) para luego enviar una señal de corriente con base a un intervalo de presión de entrada para una señal de respuesta en un intervalo de corriente; es necesario un conversor de corriente a tensión de voltaje que tome un intervalo de corriente de entrada y envíe un intervalo de voltaje de salida hacia la tarjeta, finalmente se

requiere de un conversor que funcione igual que el anterior pero de forma inversa para la señal de tensión a corriente.

- **Funcionamiento del sistema electroneumático del equipo:**

El sistema electroneumático del equipo funciona inicialmente con la señal de corriente que emite el transductor de presión hacia el conversor de corriente a voltaje, el conversor toma la señal y por medio de su circuito emite una señal de voltaje a la tarjeta, la tarjeta controladora toma la señal de tensión, esta información la procesa y la envía al conversor de voltaje a corriente, el conversor toma la señal y por medio de su circuito y emite una señal de corriente hacia la válvula proporcional para abrirla con una variación de apertura y así controlar la presión de salida estable, luego la tarjeta envía la señal de tensión a las válvulas solenoides de forma simultánea según se requiera finalmente la tarjeta debe enviar las señales necesaria para restablecer el sistema y los componentes.

El proceso indicado anteriormente tiene un orden estandarizado y dirigido por medio del código instalado en la tarjeta el cual establece en qué momento se debe ejecutar cada envío y lectura de señal, el código algorítmico instalado en la tarjeta es controlado por medio de un panel de control al momento de ser operado, este se encarga de enviar la orden por medio de señales electrónicas para ejecutar el código algorítmico que activará todos los procesos de control de los componentes del sistema electroneumático del equipo.

5.1.3 Elección de los componentes

Es necesario establecer los componentes que conforman el equipo y de acuerdo a sus características estos deben tener la capacidad de soportar las condiciones físicas establecidas en la prueba y tener un rango de desempeño confiable, es decir que estos deben estar en capacidad de soportar magnitudes físicas mayores a las usadas en las pruebas, los componentes seleccionados que cumplen con los estándares requeridos son:

- **Válvula solenoide normalmente cerrada:**
    - Referencia: AIRTAC - IP65 - CE Way - 2/2
    - Modelo : 2w050 - 10
    - Presión: 0 - 150 PSI
    - Orificio: 5 mm
    - Voltaje de alimentación: 12 VDC
        - **Cantidad : 2**
- **Válvula solenoide normalmente abierta:**
    - Referencia: AIRTAC - IP65 - CE Way - 2/2
    - Modelo : 2w050 - 10

- Presión: 0 - 150 PSI
- Orificio: 5 mm
- Voltaje de alimentación: 12 VDC

    - **Cantidad : 1**

- **Transductor de presión:**

    - Referencia: Transductor de presión (Sensor de presión)
    - Modelo : MBS 1700
    - Presión: 362.594 PSI

    - **Cantidad : 1**

- **Válvula proporcional :**

    - Referencia: Válvula Bola Dn8 1/4" Actuador Proporcional 4-20mA
    - Tamaño del producto:1 ", DN8
    - Válvula material del cuerpo: BRONCE
    - de funcionamiento: DC9V-24V
    - Señal de control:4-20mA opcional 0 a 10V o 0 a 5VDC
    - Tiempo de giro 90º: 7 Seg
    - Presión máxima de trabajo: 150 Psi
    - Corriente de trabajo:500MA
    - Tiempo de vida:100,000 ciclos
    - Material del actuador: PpO
    - Material de sello: FKM y PTFE
    - Angulo de rotacion actuador: 90°
    - Fuerza de par de torsión: 2Nm
    - Longitud del cable:0.5 M
    - Temperatura ambiente: -15°C a 60°C
    - Temperatura Fluido: 1°C a 95°C
    - Accionamiento manual: Ninguno
    - Con Indicador de apertura visual
    - Clase de protección: IP67
    - Temperatura ambiente: -15 ° a 60 °C
    - Temperatura del líquido: 1 ° a 95 °C
    - Aplicación: Agua / Aire.

    - **Cantidad : 1**

- **Módulo Conversor de corriente a voltaje:**

    - Modelo : A233
    - Señal de corriente 0-20 mA
    - Voltaje de alimentación: 0 A 5 V

    - **Cantidad : 1**

- **Módulo Conversor de voltaje a corriente:**
    - Modelo : A233
    - Señal de corriente 0-20 mA
    - Voltaje de alimentación: 0 A 5 V
        - **Cantidad : 1**
- **Módulo Conversor de corriente a voltaje:**
    - Modelo : A233
    - Señal de corriente 0-20 mA
    - Voltaje de alimentación: 0 A 3.3 V
        - **Cantidad : 1**
- **Módulo Conversor de voltaje a corriente:**
    - Modelo : A233
    - Señal de corriente 0-20 mA
    - Voltaje de alimentación: 0 A 3.3 V
        - **Cantidad : 1**
- **Relé:**
    - Características:
    - SHENLE SB
    - JQC-3FC(T73)
    - 10 A 125 VAC
    - 10 A 28 VDC
    - DC 5V
        - **Cantidad : 3**
- **Tarjeta controladora Launchpad Texas Instrument**
    - MSP ULP FRAM basado en tecnología MSP430FR2355 MCU de 16 bits
    - Tecnología Energy Trace disponible para depuración de energía ultra baja
    - Kit de desarrollo LaunchPad de 40 pines estándar que aprovecha el ecosistema del módulo complementario BoosterPack
    - Sonda de depuración eZ-FET integrada
    - 2 botones y 2 LED para la interacción del usuario
    - Sensor de luz ambiental para la demostración de la experiencia lista para usar
    - Conector Grove para sensores Grove externos
        - **Cantidad : 1**
- **Tarjeta controladora Arduino Mega 2560**
    - Microcontrolador: ATmega2560

- Voltaje Operativo: 5V
- Tensión de Entrada: 7-12V
- Voltaje de Entrada(límites): 6-20V
- Pines digitales de Entrada/Salida: 54 (de los cuales 14 proveen salida PWM)
- Pines análogos de entrada: 16
- Corriente DC por cada Pin Entrada/Salida: 40 mA
- Corriente DC entregada en el Pin 3.3V: 50 mA
- Memoria Flash: 256 KB (8 KB usados por el bootloader)
- SRAM: 8KB
- EEPROM: 4KB
- Clock Speed: 16 MHz

    - **Cantidad : 1**

- **Adaptador de 115 Vac a 12 Vdc**

    - Voltaje operativo 110v a 12V DC,
    - Entrada: 100-240V AC.
    - frecuencia 50-60 Hz.
    - Salida: 12V 1A.

    - **Cantidad : 1**

5.1.4 Diagrama del sistema neumático y electroneumático del equipo

---

- **Descripción General:**

A Partir de la necesidades de operación del fluido se establece por medio de un diagrama electroneumático la interacción que deben tener los componentes electroneumáticos con las líneas neumáticas de presión y cómo deben ir conectados entre sí para su funcionamiento, es decir que los dos sistemas deben funcionar sincronizadamente claramente el mando del sistema del fluido es controlado electrónicamente, todo el control interno del sistema neumático depende del sistema electrónico y las instrucciones que brinda la tarjeta a los componentes para la liberación o retención del fluido según se requiera.

- **Distribución eléctrica del equipo:**

El equipo cuenta con una alimentación de 115 Vac como fuente principal, la cual ingresa al adaptador que luego la transforma a 12Vdc, este voltaje directo energiza los componentes electroneumáticos cómo lo son la válvula proporcional, las válvulas tipo solenoide on /off, el transductor de presión y la válvula de alivio, la tarjeta de control es energizada con una tensión de 5Vdc que salen de la computadora de control, esta tarjeta es el cerebro del equipo, contiene la información algorítmica que se debe ejecutar según lo indique

la computadora de control, la tarjeta recibe y envía señales analógicas de 0 a 5Vdc o de 0 a 3.3Vdc del transductor de presión y para la válvula proporcional respectivamente, para estos dos componentes es necesario un conversor de voltaje a corriente y un conversor de corriente a voltaje ya que el transductor envía una señal de 4 - 20 mA según la presión sensada y la válvula recibe de 4 a 20mA para la apertura proporcional, finalmente la tarjeta distribuye 5Vdc o 3.3Vdc hacia los tres relés que activan las líneas de tensión de 12Vdc de las válvulas on/off , está tensión de salida o entrada de 5Vdc o 3.3Vdc depende de la tarjeta.

- **Distribución del sistema neumático del equipo:**

La distribución neumática del equipo cuenta con una presión inicial dada por un flujo de aire o nitrógeno de 120 a 140 PSI,el cual es transmitido a la primera fase de recorrido del fluido , para este primer recorrido del fluido, este mantiene su presión inicial y la presión es sensada por el transductor de presión y leída por un manómetro, luego la válvula proporcional abre el paso de la línea neumática y el fluido pasa a la segunda fase del sistema neumático, allí la presión es leída por un manómetro secundario para verificar que la presión haya disminuido después de pasar por la válvula proporcional , luego el sistema se divide en tres líneas dos salidas controladas con una válvula solenoide normalmente cerrada para cada salida y la tercera línea es una salida controlada con una válvula solenoide normalmente abierta, estas tres válvulas serán activadas y desactivadas para el control y la salida de fluido según sea necesario.

- **Prueba 1 :**

Está prueba es una configuración de ensamble de los componentes electrónicos del equipo controlados por una **Tarjeta Arduino mega 2560.**

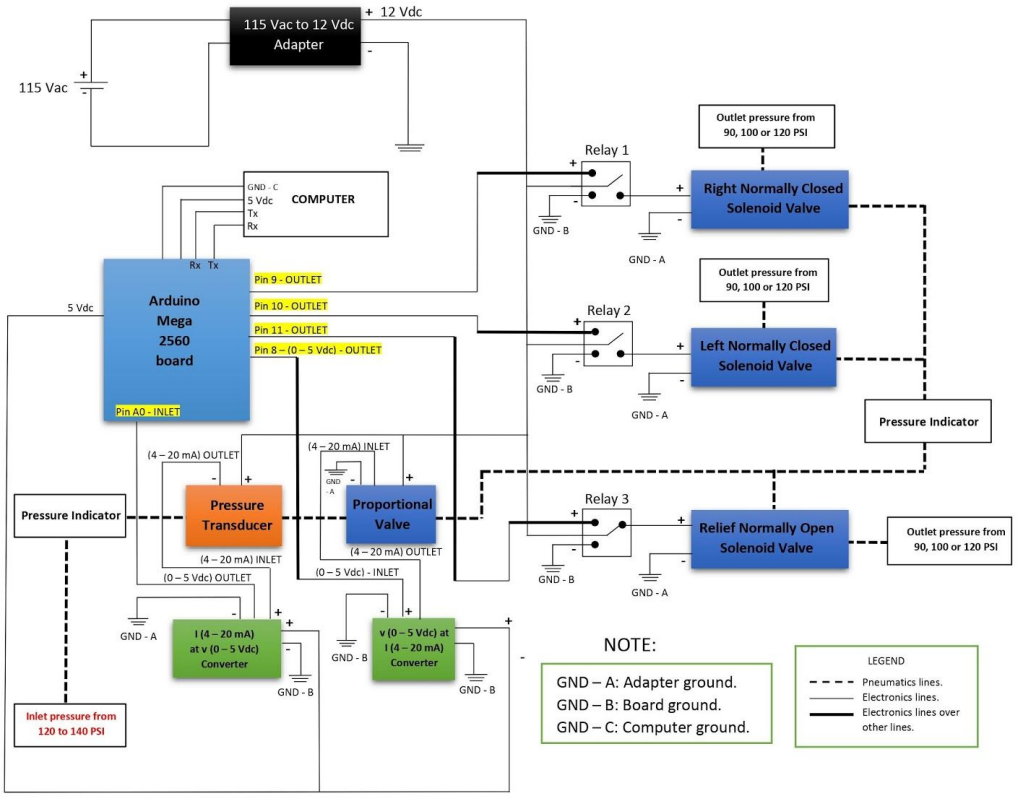

Figura 3.26 Diagrama General Electroneumático de simulación del equipo de apoyo en tierra del sistema Rudder Boost King Air Series 200. Fuente: autores.

- **Observaciones:** Las condiciones eléctricas de la prueba muestran que la tarjeta opera con una tensión de 0 - 5 Vdc para salidas y entradas programadas de señal lo cual es importante tener claro para la calibración de los conversores.

- **Prueba 2:**

Está prueba es una configuración de ensamble de los componentes electrónicos del equipo controlados por una **Tarjeta Launchpad Texas Instrument MSP- EXP430G2ET** con un microcontrolador compatible **MSP430G2553.**

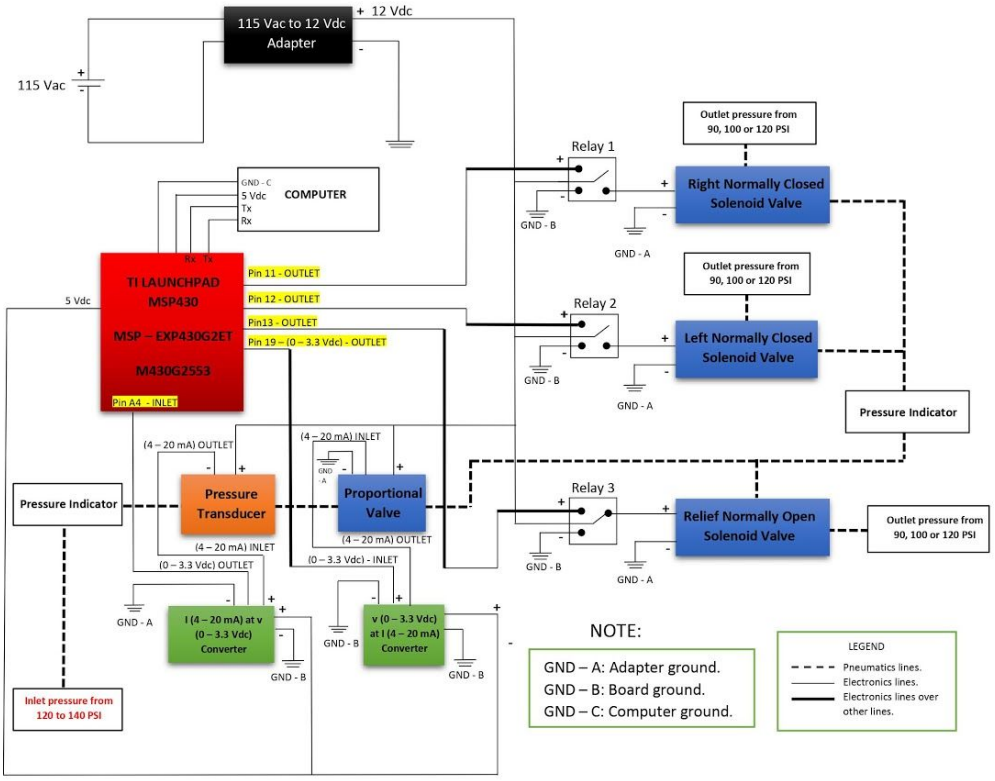

Figura 3.27 Diagrama General Electroneumático de simulación del equipo de apoyo en tierra del sistema Rudder Boost King Air Series 200. Fuente: autores.

- **Observaciones:** Las condiciones eléctricas de la prueba muestran que la tarjeta opera con una tensión de 0 - 3.3Vdc para salidas y entradas programadas de señal, esto implica que los conversores deben adecuarse a esta variación de voltaje.

5.2 ALGORITMO DE CONTROL DEL EQUIPO

Para la secuencia lógica de activación de los componentes y para el control del equipo en general se pretende determinar un algoritmo de control que muestre las instrucciones principales necesarias para la prueba de chequeo ejecutadas por el operador del equipo desde un panel de control, a partir de este algoritmo de control la programación es la siguiente tarea para la automatización del equipo, ya que esta lleva a cabo el lenguaje leído por la tarjeta para la interacción sincrónica de señales de tensión, corriente y comunicación entre el panel de control, la tarjeta y los componentes.

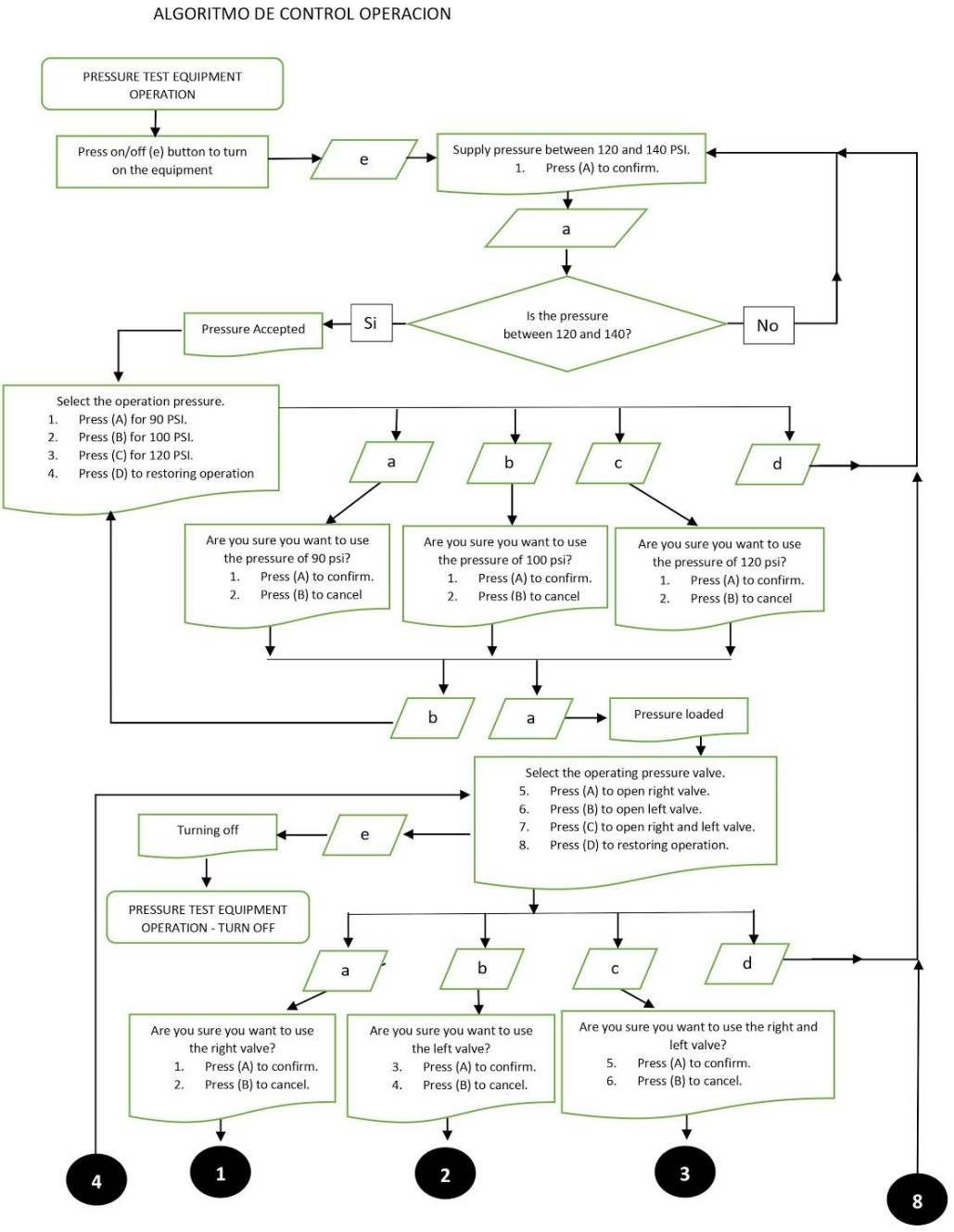

Figura 3.28 (a) Diagrama de flujo de secuencia lógica de activación de los componentes y para el control del equipo de apoyo en tierra del sistema Rudder Boost King Air Series 200:Fuente autores.

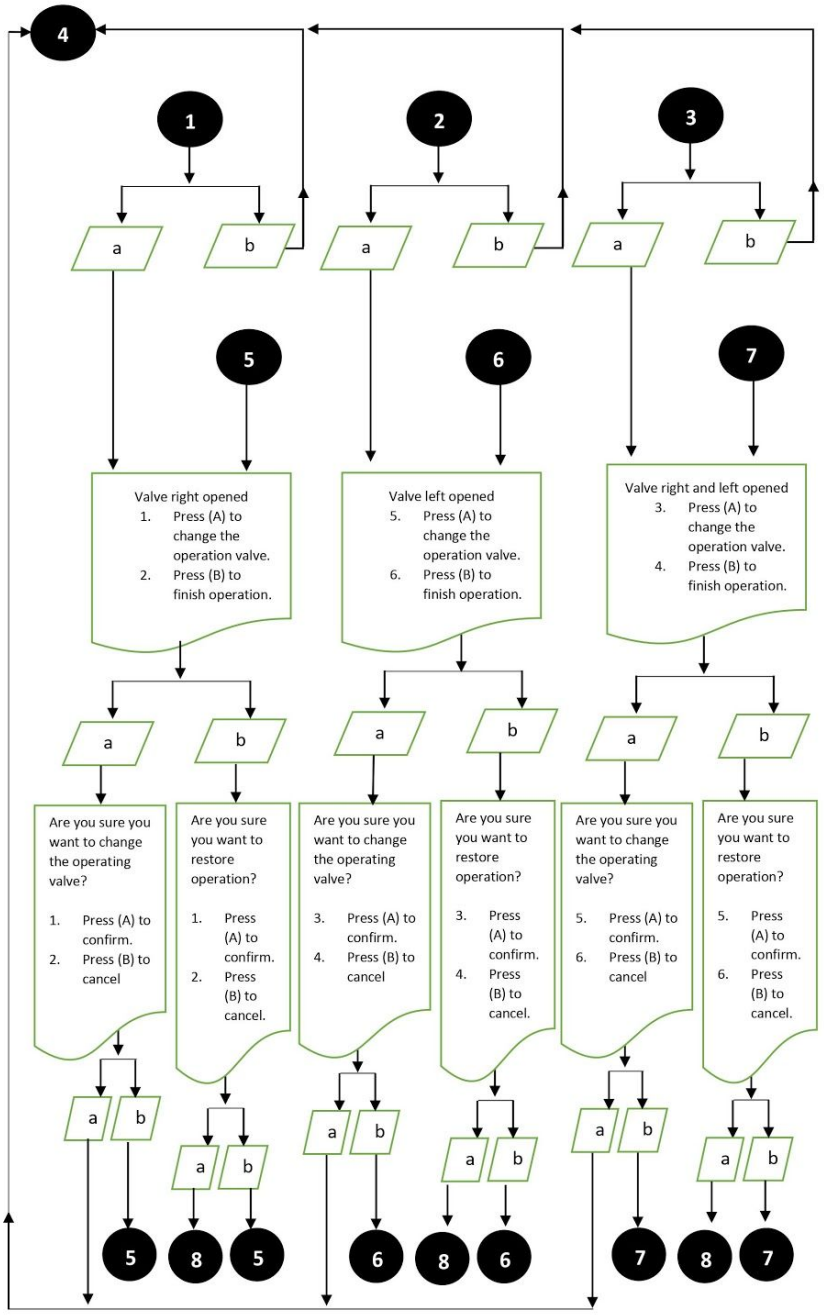

Figura 3.28 (b)Diagrama de flujo de secuencia lógica de activación de los componentes y para el control del equipo de apoyo en tierra del sistema Rudder Boost King Air Series 200:Fuente de autores.

Con base en las instrucciones dadas en el algoritmo se tiene la secuencia lógica en la cual deben activarse y desactivarse los componentes para el control se procede con la creación del entorno del panel por medio del software Processing quien tendrá comunicación con

arduino y/o energía para la ejecución del código programado.

5.3 CÁLCULOS Y GRÁFICAS

5.3.1 Voltaje de salida de apertura de válvula proporcional

Para determinar la señal de voltaje de salida hacia la válvula proporcional en el programa se necesita una conversión lineal que indique el cambio de voltaje hacia esta válvula, ya que la presión inicial varía, es decir que la presión inicial se estandariza cómo la presión máxima que tiene la válvula proporcional al momento de estar totalmente abierta y si la salida de la válvula proporcional tiene que ser siempre fija para tres opciones diferentes es necesario una ecuación que tome el valor de intervalo de presión en un intervalo de voltaje para luego aplicar la fórmula y tomar un nuevo valor de voltaje para la presión que finalmente debe salir por la válvula proporcional, para entender mejor este principio a continuación se muestra de forma matemática la variación de este voltaje.

- **Variables:**

$V1 \rightarrow Voltaje\ de\ entrada\ (Voltaje\ dada\ por\ el\ conversor\ corriente\ -\ voltaje)$.

$V2 \rightarrow Voltaje\ de\ salida\ de\ la\ tarjeta\ hacia\ el\ conversor\ de\ la\ válvula$.

$Vmax \rightarrow Voltaje\ máximo\ de\ intervalo\ de\ variación\ de\ voltaje$.

$P1 \rightarrow Presión\ inicial\ (Presión\ leída\ por\ el\ sensor)$.

$P2 \rightarrow Presión\ de\ salida\ de\ la\ válvula\ proporcional$.

$Pmax\ 1 \rightarrow Presión\ máxima\ de\ operación\ del\ sensor$.

$Pmax\ 2 \rightarrow Presión\ máxima\ de\ operación\ de\ la\ válvula\ proporcional$.

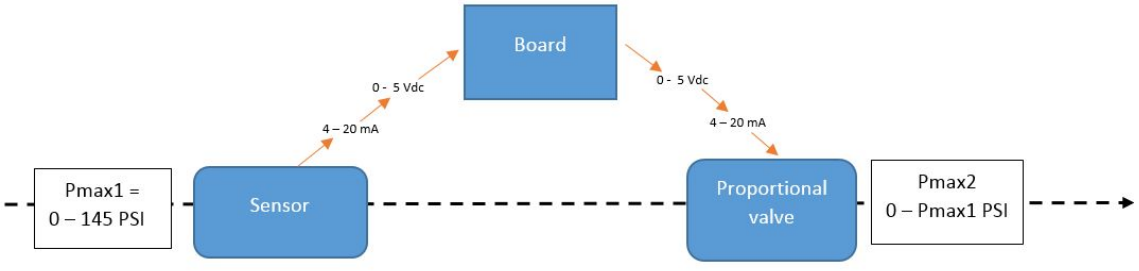

Figura 3.29 Diagrama de entrada y salida de las señales electrónicas del sensor, la tarjeta y

la válvula proporcional. Fuente: autores.

Para determinar el valor de presión de entrada se tiene una variación de voltaje en relación con la variación de presión, es decir que el valor máximo de presión leído por el sensor, tendrá cómo resultado el envío del máximo valor de voltaje, por lo tanto se tiene la siguiente ecuación:

$$\frac{P1}{Pmax\ 1} = \frac{V1}{V\,max} \tag{1}$$

Cómo condición inicial se conoce el valor del voltaje máximo, la presión máxima y el voltaje de entrada a la tarjeta, por lo tanto se despeja el valor de la presión inicial (P1) para hallar la presión máxima que tendrá la válvula proporcional así,

$$P1 = \frac{V1\ \ Pmax\ 1}{V\,max} = Pmax\ 2 \tag{2}$$

después de obtener la presión máxima en función del voltaje inicial se debe identificar cómo cambia el voltaje de tarjeta hacia la válvula proporcional.

$$\frac{Pmax\ 2}{P2} = \frac{V\,max}{V\,2} \tag{3}$$

Para simplificar los valores de presión y obtener la ecuación en función de valores de voltaje y valores conocidos se despeja la presión máxima de la válvula para luego igualarla con la ecuación (2) así:

$$Pmax\ 2 = \frac{V\,max\ P2}{V\,2} \tag{4}$$

Se procede entonces a igualar la ecuación (2) y (4) para determinar la función estándar de variación de voltaje,

$$\frac{V1\ \ Pmax\ 1}{V\,max} = \frac{V\,max\ P2}{V\,2} \tag{5}$$

Simplificando la ecuación quedaría de la siguiente forma:

$$V\,2 = \frac{V\,max * V\,max\ P2}{V\,1 * Pmax\ 1} \tag{6}$$

Finalmente se tienen tres diferentes casos para tres presiones de salida de la válvula proporcional que son estables para tres valores diferentes que en este caso serían los valores de 90, 100 y 120 PSI cómo presión de salida de la válvula,

- CASO 1: Presión de salida (P2 = 90 PSI)

$$V\,2 = \frac{V\,max^{\,2}\ \ (90)}{V\,1 * Pmax\ 1} \tag{7}$$

- CASO 2: Presión de salida (P2 = 100 PSI)

$$V2 = \frac{V\,max^{2}\ \ (100)}{V1 * P\,max\ 1} \tag{8}$$

- CASO 3: Presión de salida (P2 = 120 PSI)

$$V2 = \frac{V\,max^{2}\ \ (120)}{V1 * P\,max\ 1} \tag{8}$$

NOTA : Para la placa que tiene salidas y entradas de 3.3 voltios, este seria el voltaje máximo a comparación de la placa que maneja 5 Voltios, en este caso seria su valor máximo.

5.3.2 Gráficas - Voltaje de salida de apertura de válvula Vs voltaje

Para la determinación de la gráfica que demuestra el comportamiento del cambio de voltaje de salida hacia el conversor de la válvula proporcional, con respecto al cambio de presión de entrada se toma la ecuación (8) hallada anteriormente en deducción matemática simple, en la Figura 3.30 (a) la gráfica está dada para una tarjeta que cumpla con las tensión de operación de 0 - 5 voltios y en la Figura 3.30 (b) la gráfica está dada para una tarjeta que cumpla con un voltaje operativo de 0 a 3.3 Vdc.

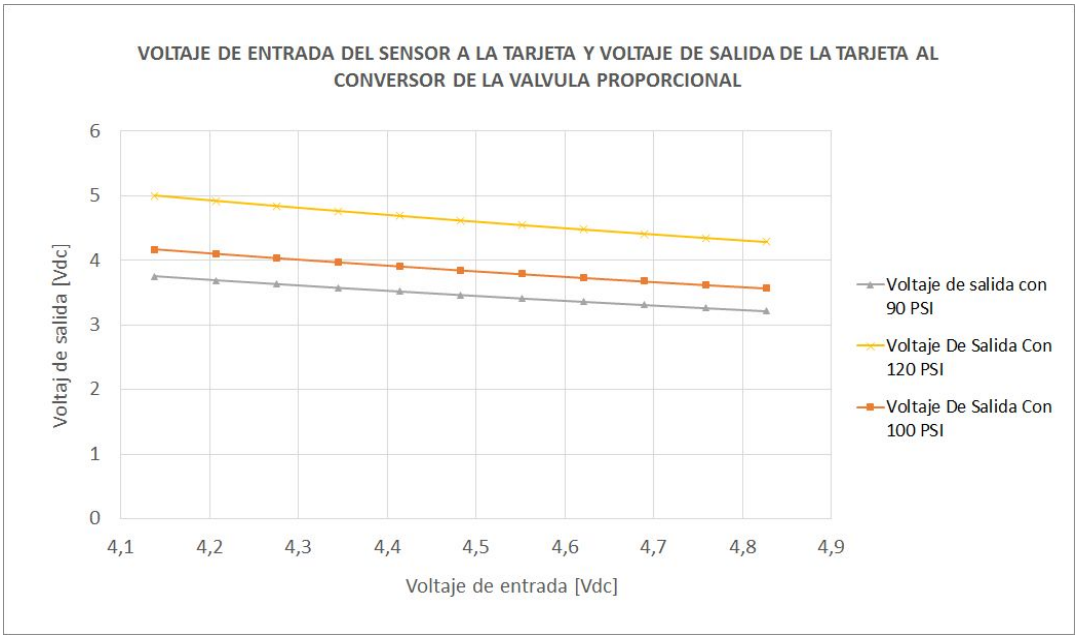

Figura 3.30 a) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente de de autores.

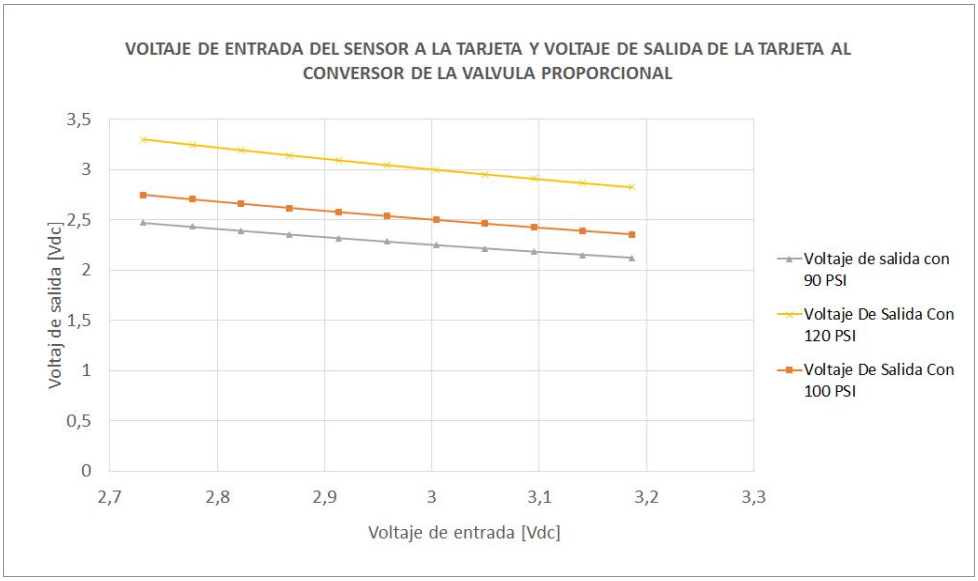

Figura 3.30 b) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: autores.

5.4 DESARROLLOS DE LOS CÓDIGOS DE LENGUAJE

Cómo resultado de la creación de los códigos de programación, se tiene un código que va instalado en la tarjeta controladora Arduino y/o LaunchPad Texas instrument con el fin de tener información de pruebas en estos dos prototipos y ver el alcance de los mismos, la edición del código tiene diferencias en los valores de los pines de entrada y salida y en el voltaje operativo; por otra parte se obtiene un código en el software processing para el desarrollo del panel de control operativo.

5.4.1 Desarrollo del código para la tarjeta Arduino Mega 2560

El código desarrollado establece que la tarjeta tiene un pin de entrada y otro de salida de voltaje analogico y alimenta a los componentes del equipo por medio de tres pines, esto se aprecia en la siguiente descripción de configuración de los pines de la tarjeta.

- **Pin 8:** Salida analogica de voltaje de 0 - 5 Vdc con el objetivo de abrir o cerrar la válvula proporcional.

- **Pin 9:** Salida digital de voltaje de 0 o 5 Vdc para la apertura o cierre de la válvula de alivio del sistema.
- **Pin 10:**Salida digital de voltaje de 0 o 5 Vdc para la apertura o cierre de la válvula derecha del sistema.
- **Pin 11:**Salida digital de voltaje de 0 o 5 Vdc para la apertura o cierre de la válvula izquierda del sistema.
- **Pin A0:** Entrada analogica de voltaje de 0 - 5 Vdc con el objetivo de tomar el valor de voltaje para identificar la presión inicial.

Para la conversión de voltaje de entrada como el voltaje de salida se usan los siguientes valores:

$$V1 \rightarrow Voltaje\ de\ entrada\ =\ Dado\ por\ la\ entrada\ de\ presión$$

$$Vmax \rightarrow Voltaje\ máximo\ de\ intervalo\ de\ variación\ de\ voltaje.\ =\ 5\ Vdc$$

$$Pmax\ 1 \rightarrow Presión\ máxima\ de\ operación\ del\ sensor.\ = 145\ PSI$$

$$P2 \rightarrow Presión\ de\ salida\ de\ la\ válvula\ proporcional.\ =\ Obtenido\ a\ partir\ de\ la\ ecuación$$

El código realizado se puede apreciar en el **Apéndice C,** el desarrollo muestra cada uno de los pasos algorítmicos que cumplen con las condiciones de control del equipo.

5.4.2 Desarrollo del código para la tarjeta LaunchPad MSP-EXP430G2ET

El código desarrollado mantiene el mismo principio del código anterior, es decir que contiene un pin de entrada y otro de salida de voltaje analogico y alimenta a los componentes del equipo por medio de tres pines, la diferencia es la referencia de los pines y el voltaje de operación de la tarjeta, esto se aprecia en la siguiente descripción de configuración de los pines de la tarjeta.

- **Pin 11:** Salida digital de voltaje de 0 o 3.3 Vdc para la apertura o cierre de la válvula derecha del sistema.
- **Pin 12:** Salida digital de voltaje de 0 o 3.3 Vdc para la apertura o cierre de la válvula izquierda del sistema.
- **Pin 13:** Salida digital de voltaje de 0 o 3.3 Vdc para la apertura o cierre de la válvula de alivio del sistema
- **Pin 19:** Salida analogica de voltaje de 0 - 3.3 Vdc con el objetivo de abrir o cerrar la válvula proporcional.
- **Pin A4:** Entrada analogica de voltaje de 0 - 3.3 Vdc con el objetivo de tomar el valor de voltaje para identificar la presión inicial.

Para la conversión de voltaje de entrada como voltaje de salida se usan los siguientes

valores:

$V1 \rightarrow Voltaje\ de\ entrada\ =\ Dado\ por\ la\ entrada\ de\ presión$

$V\,max \rightarrow Voltaje\ máximo\ de\ intervalo\ de\ variación\ de\ voltaje.\ =\ 3.3\ V\,dc$

$P\,max\ 1 \rightarrow Presión\ máxima\ de\ operación\ del\ sensor.\ =\ 145\ PSI$

$P2 \rightarrow Presión\ de\ salida\ de\ la\ válvula\ proporcional.\ =\ Obtenido\ de\ la\ ecuación$

El código realizado se puede apreciar en el **Apéndice D,** el desarrollo muestra cada uno de los pasos algorítmicos que cumplen con las condiciones de control del equipo.

5.4.3 Desarrollo del código para el panel de control del equipo

---

Con la plataforma de Processing la configuración del panel de control se crea a partir de un código, este código brinda las instrucciones de leer y enviar información por medio de la comunicación serial, está se configura al inicio del código, luego contiene unas instrucciones geométricas y gráficas para establecer el contorno del panel de control, desde allí puede apreciarse el proceso de operación y el algoritmo de control programado para el fin determinado.

El código realizado se puede apreciar en el **Apéndice D.**

5.5 PANEL DE CONTROL DEL EQUIPO

A partir de la configuración del código y la ejecución del mismo se puede apreciar cómo desde el panel de control se maneja la codificación realizada en las tarjetas ya programadas, en la *Figura 3.31 a) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente de de autores* se muestra el panel estandarizado de control del equipo, desde allí se controlan las señales que se van a emitir de la tarjeta hacia los componentes de forma sencilla, para el operador se aprecia que es una plataforma fácil de manejar, esto también hace que la prueba de chequeo a realizar sea muy sencilla y práctica.

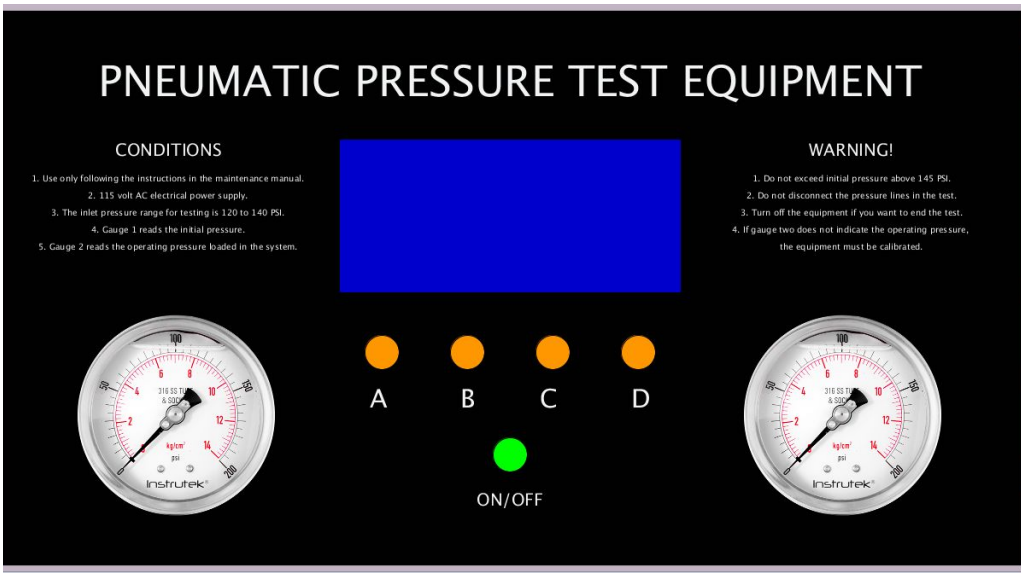

Figura 3.31 a) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional.Fuente:de de autores.

El panel de control está conformado por una pantalla que va a mostrar la información necesaria para la operación del equipo, cuatro botones de selección de opciones indicativas (A, B, C y D) a medida que se vaya mostrando cada instrucción y finalmente un botón para el encendido y apagado del panel y el equipo.

Este panel lleva información adicional de condiciones y advertencias para que el operador tenga conocimiento de los límites de operación del equipo y las condiciones que debe tener claras antes de proceder con la prueba.

5.6 PRUEBAS DE FUNCIONAMIENTO DEL PROGRAMA DE CONTROL

Para comprobar el correcto funcionamiento del sistema se debe llevar a cabo un chequeo de señales electrónicas con un multímetro en los pines de la tarjeta para cada operación sucesiva ejecutada desde el panel de control, este procedimiento se muestra a continuación por medio de una sucesión de operaciones secuenciales estandarizadas para el cumplimiento de la prueba de chequeo.

5.6.1 Operaciones del equipo para la prueba

- **Operación 1:** Encendido del equipo por medio del botón on/off.
    - **Pin 8 (Salida - Válvula proporcional)** = 0,00 Vdc
    - **Pin 9 (Salida - Válvula de alivio)** = **4,88 Vdc**
    - **Pin 10 (Salida - Válvula derecha)** = 0,00 Vdc
    - **Pin 11 (Salida - Válvula izquierda)** = 0,00 Vdc
    - **pin A0 (Entrada -)** = 0,00 Vdc

En esta primera parte del chequeo de funcionamiento del equipo, la válvula proporcional está cerrada, la válvula de alivio está energizada y se cierra ya que es una válvula normalmente abierta, las dos válvulas derecha e izquierda no están energizadas y están cerradas ya que son válvulas normalmente cerradas, por último, la señal de entrada del pin A0 tiene una tensión de 0,0 Vdc que indican que hay que inyectar presión en el rango aceptable de presión.

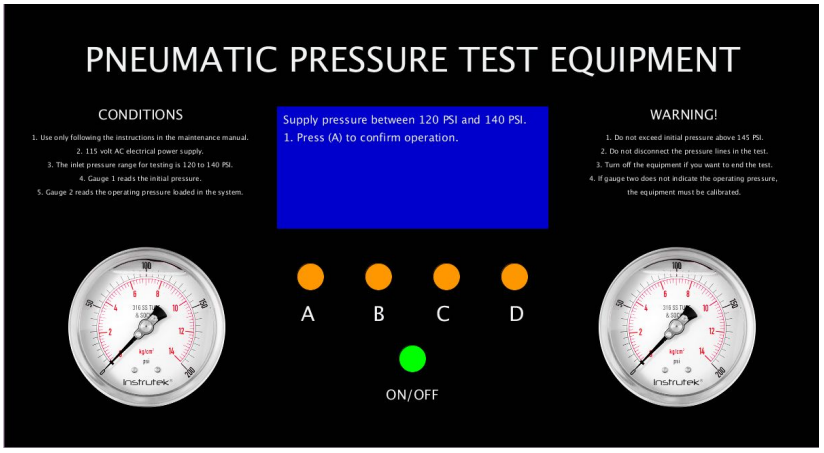

Figura 3.31 b) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: autores.

- **Operación 2:** Se inyecta presión de 120 a 140 PSI luego se oprime el botón (A). para verificar que la presión inicial está dentro del rango aceptable.
    - **Pin 8 (Salida - Válvula proporcional)** = 0,00 Vdc
    - **Pin 9 (Salida - Válvula de alivio)** = **4,88 Vdc**
    - **Pin 10 (Salida - Válvula derecha)** = 0,00 Vdc
    - **Pin 11 (Salida - Válvula izquierda)** = 0,00 Vdc
    - **pin A0 (Entrada -)** = **4,5 Vdc**.

Aparece una señal de entrada por el pin A0, por medio de la fórmula obtenida en el cálculo matemático la presión de entrada es de aproximadamente 130 PSI y cumple con el rango de operación de presión.

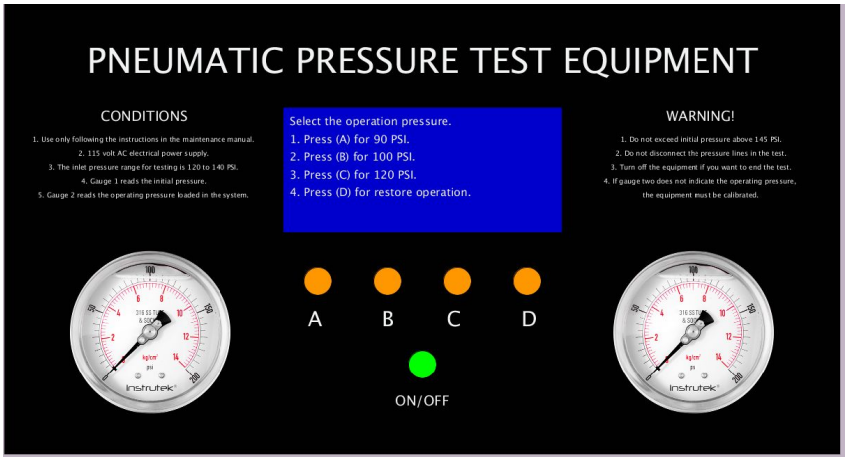

Figura 3.31 c) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional.Fuente: de autores.

- **Operación 3:** Se oprime el botón (A) para activar la apertura de la válvula proporcional para una salida de presión de 90 PSI
    - **Pin 8 (Salida - Válvula proporcional)** = **3,4 Vdc**
    - **Pin 9 (Salida - Válvula de alivio)** = **4,88 Vdc**
    - **Pin 10 (Salida - Válvula derecha)** = 0,00 Vdc
    - **Pin 11 (Salida - Válvula izquierda)** = 0,00 Vdc
    - **pin A0 (Entrada -)** = **4,5 Vdc**

La válvula proporcional cumple con la apertura adecuada según el voltaje de lectura para una salida de presión de 90 PSI, esta conversión está establecida en la *Figura 3.30 a) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente de de autores.* obtenida a partir de la fórmula de relación de voltajes.

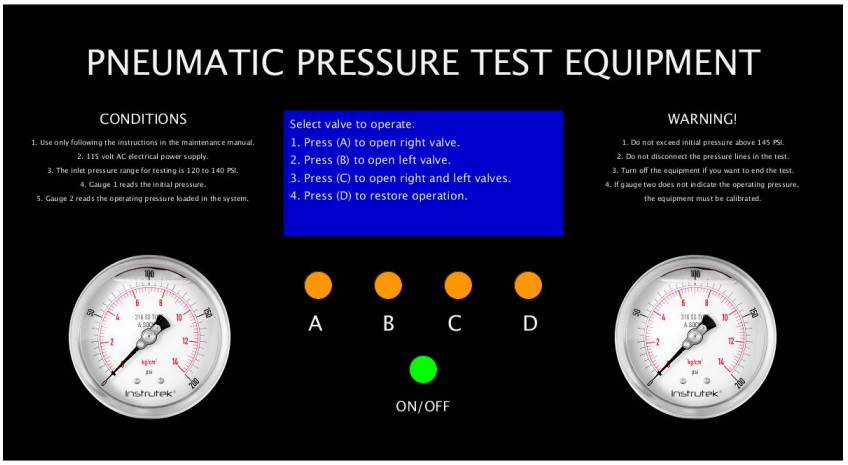

Figura 3.31 d) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional: Fuente de de autores.

- **Operación 4:** Se oprime el botón (A) para activar la válvula derecha.
    - **Pin 8 (Salida - Válvula proporcional)** = **3,4 Vdc**
    - **Pin 9 (Salida - Válvula de alivio)** = **4,88 Vdc**
    - **Pin 10 (Salida - Válvula derecha)** =**4,88 Vdc**
    - **Pin 11 (Salida - Válvula izquierda)** = 0,00 Vdc
    - **pin A0 (Entrada -)** = **4,5 Vdc**

La señal de voltaje para activar la válvula derecha se ejecuta correctamente.

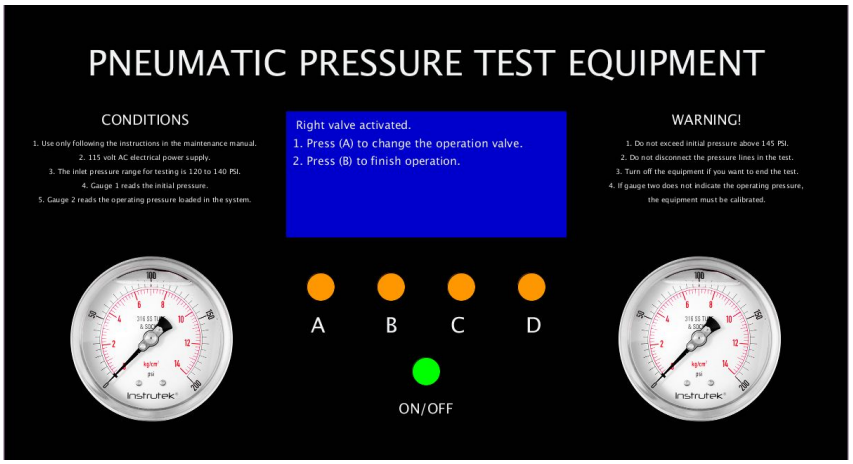

Figura 3.31 e) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: de de autores.

- **Operación 5:** Se oprime el botón (A) para cerrar la válvula y regresar al menú de apertura de válvulas.
    - **Pin 8 (Salida - Válvula proporcional)** = **3,4 Vdc**
    - **Pin 9 (Salida - Válvula de alivio)** = **4,88 Vdc**
    - **Pin 10 (Salida - Válvula derecha)** =0,00 Vdc
    - **Pin 11 (Salida - Válvula izquierda)** = 0,00 Vdc
    - **pin A0 (Entrada -)** = **4,5 Vdc**

La señal de voltaje para la apertura de la válvula derecha se desactiva correctamente.

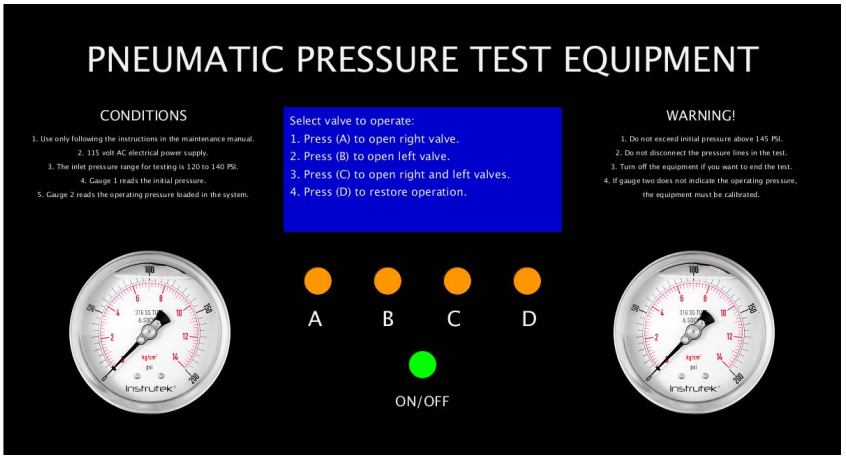

Figura 331 f) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: de de autores.

- **Operación 6:** Se oprime el botón (D). para desactivar las señales de voltaje y apagar el equipo
    - **Pin 8 (Salida - Válvula proporcional)** = 0,00 Vdc
    - **Pin 9 (Salida - Válvula de alivio)** = 4,88 Vdc
    - **Pin 10 (Salida - Válvula derecha)** = 0,00 Vdc
    - **Pin 11 (Salida - Válvula izquierda)** = 0,00 Vdc
    - **pin A0 (Entrada -)** = 4,5 Vdc

La válvula proporcional está cerrada, la válvula de alivio se cierra ya que es una válvula normalmente abierta, las dos válvulas derecha e izquierda están cerradas ya que son válvulas normalmente cerradas, de esta forma queda restablecida la operación y se apaga el monitor del panel.

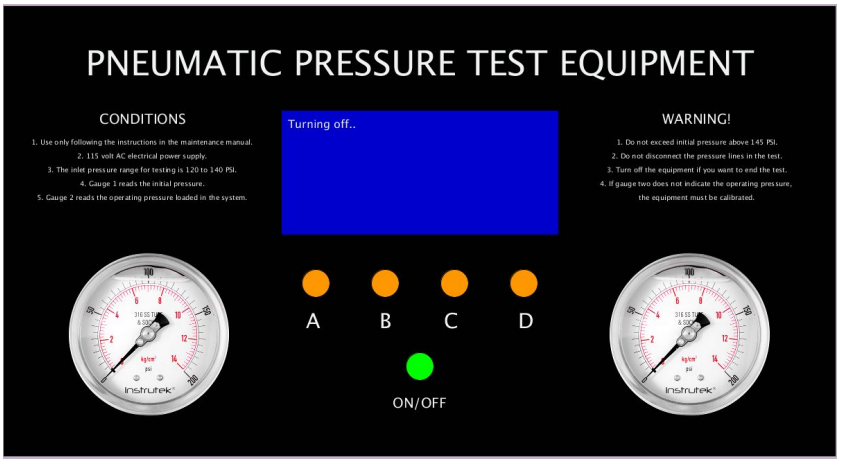

Figura 331 g) Diagrama de entrada y salida de las señales electrónicas del sensor , la tarjeta y la válvula proporcional. Fuente: de de autores.

5.6.2 Observaciones de las pruebas de funcionamiento

- La anterior prueba es solo de una ruta de opciones del programa, sin embargo para este código y para las dos tarjetas programadas haciendo una revisión minuciosa de las señales, estas se envían correctamente en cada instrucción asignada de manera exitosa.
- La fórmula asignada para la apertura de la válvula proporcional del cambio de voltaje de salida con respecto a la variación de la salida de presión se deduce y se comporta de forma lineal, esto es solo para garantizar el correcto funcionamiento del programa, es decir que la fabricación del equipo tiene que llevar a cabo pruebas neumáticas y mirar el cambio de presión con respecto al cambio de voltaje para configurarla la fórmula en el programa del control.

Capítulo 7

CONCLUSIONES

- El sistema Rudder Boost es un sistema redundante del avión que se activa en caso de emergencia, su función es compensar la estabilidad lateral del avión mediante el Rudder por medio de una compensación neumática para estabilizar el avión en caso de falla de alguno de sus dos motores, por esta razón es importante hacer pruebas de chequeo de funcionamiento para este sistema y garantizar su correcto funcionamiento constantemente mediante un equipo de pruebas, se opta por un equipo de apoyo en tierra ya que es la forma más efectiva, a comparación de la segunda prueba de chequeo posible para éste sistema donde se deben iniciar los motores, si se usa un equipo de apoyo en tierra los costos y tiempos operativos se reducen de forma significativa.

- Antes de proceder con la programación de la tarjeta, se debe tener claro por medio de un diagrama el cómo los componentes que conforman el equipo deben estar conectados entre sí para luego identificar el proceso lógico de funcionamiento de estos por medio de un algoritmo. El uso de algoritmos es fundamental para el entendimiento del proceso de control operativo del sistema que se desea automatizar, por eso este se recomienda usar en procesos de mantenimiento que cumplan un ciclo de actividades cómo lo es la prueba de chequeo del sistema Rudder Boost.

- El proceso de programación depende en gran medida del conocimiento del lenguaje y entorno del software a utilizar ya que sin estas habilidades previas no es posible generar un lenguaje de programación que sea efectivo y que de soluciòn a las necesidades requeridas, por lo tanto es recomendable tener el estudio previo de la sintaxis y el lenguaje de programación antes de la creación del código del programa de control. Para este caso el lenguaje de programación realizado en Arduino, Energía y Processing para el equipo de apoyo en tierra cumple con las exigencias y brinda las instrucciones precisas de acuerdo al algoritmo de control previamente creado.

- A partir de un interfaz gráfico programando en Processing es posible visualizar el control de operación del equipo, por medio de su operación se garantiza una adecuada sincronización del entorno de programaciòn y los componentes, también se determina el óptimo desempeño operacional de todo el sistema por medio de pruebas de lectura de señales eléctricas tomadas en los pines de distribución electrónica de las tarjetas, obteniendo altos estándares de confiabilidad y aseguramiento de calidad en la pruebas de chequeo funcional de sistema Rudder Boost en el Beechcraft King Air Series 200 que debe realizar el equipo de apoyo en tierra.

BIBLIOGRAFÍA

[1] Aircraft Maintenance manual Beechcraft king air 200 and B200. Emisión: Febrero 27 de 1998. Revisión: abril 28 del 2006.

[2] Aero Expo. Tester de compresion de motor portátil, en línea. Ubicacion: https://www.aeroexpo.online/es/prod/bauer-inc/product-183345-24173.html. Recuperado el 14 de mayo del 2020.

[3] Aero Expo tester de compresion / de motor / para la aeronáutica / portátil, en línea ubicacion://www.aeroexpo.online/es/prod/howell-instruments/product-184407-32332.html Recuperado el 16 de mayo del 2020.

[4] AGUAMARKET. Válvula reductora de presión de acción directa, en línea. Ubicacion: https://www.aguamarket.com/productos/productos.asp?producto=6278. Recuperado el 18 de mayo del 2020.

[5]ArduinoandProcessingWorkshop.https://cla.purdue.edu/academic/rueffschool/ad/etb/resources/Processing_and_Arduino.pdf.

[6] Buñay Juan Carlos (Previo a la obtención del título de ingeniero industrial mención mantenimiento milagro), República del Ecuador Enero del 2011,,137 páginas (Diseño De Un Equipo De Apoyo En Tierra Para El Relleno De Nitrógeno A Baja Y Alta Presión Que Será Utilizado En Aviones De Combate De La Base Aérea De Taura) Universidad estatal de milagro unidad académica ciencias de la ingeniería.

[7] Cabrera Dany Alberto**.** Escuela Politécnica Nacional (Proyecto Previo a la obtención del título de Tecnólogo Electromecánico Quito 2009 152 páginas Diseño y construcción de un módulo didáctico para el marcado de piezas en serie que permita mejorar la enseñanza aprendizaje en el área de neumática y control Escuela Politécnica Nacional.

[8] CORTÉS, Fernando. MONJARAZ, Jaime. Arduino: Aplicaciones en robótica, mecatrónica e ingenierías. ISBN: 9789586829762. México: Alfaomega, 2015. 428 páginas.

[9] CREUS, Antonio. Iniciación a la Aeronáutica.

[10]Como hacer una bibliografía con normas Icontec https://normasicontec.co/bibliografia//.

[11] DIRECTINDUSTRY. Regulador reductor de presión neumática para aire comprimido.

[12] Federal Aviation Administration. Airport Coperative Research Program Group support equipment (GSE) Emission reduction strategies Inventory and Tutorial (https)://books.google.com.co/books?id=EwcRp_jTHhoC&printsec=frontcover&dq=ground+equipment+aviation&hl=es&sa=X&ved=0ahUKEwiCkb6o_9ToAhVkTd8KHdM5

CMYQ6AEIMDAB#v=onepage&q=ground%20equipment%20aviation&f=false.

[13] FESTO Válvula neumática ubicacion en linea https://www.festo-didactic.com/int-en/ Recuperado el 18 de mayo del 2020.

[14] Gutiérrez José Tutorial Arduino Recuperado el 17 de mayo del documento original Arduino Notebook: A Beginner 's Reference Written and compiled by Brian W. Evans With information or inspiration taken from: http://www.arduino.cc Published: https://arduinobot.pbworks.com/f/Manual+Programacion+Arduino.pdf.

[15] HNSA. Válvulas reguladoras de caudal en línea. Ubicación: http://www.hnsa. com.co/valvulas-reguladoras-de-caudal-en-linea/. Recuperado el 18 de mayo del 2020.

[16] HIDROTEN Valvulas anti retorno de bola ball en linea https://hidroten.com/es/familia/valvulas-valvulas-antirretorno-de-bola-ball Recuperado el 19 de mayo del 2020.

[17]laurence.com.ar/artes/comun/Processing_un_lenguaje_al%20alcance_de_todos.pdfhttps://cla.purdue.edu/academic/rueffschool/ad/etb/resources/Processing_and_Arduino.pdf.

[18] LOZANO, Daniel. Arduino Práctico. ISBN: 9788441538382. Madrid: Anaya Multimedia, 2017. 327 páginas (https)://www-ebooks7-24-com.biblioteca.libertadores.edu.co/?il=9081&pg=19 .

[19] MARGOLIS, Michael. Make an arduino - controlled robot. ISBN: 1449344372; 9781449344375. Sebastopol, Calif. : O'Reilly, 2013. 238 páginas.

[20] Manuel A. Perales, Federico J. Barrero, Sergio L. "Toral Análisis comparativo de distintas plataformas para la enseñanza de Sistemas Electrónicos Digitales" https://core.ac.uk/download/pdf/51405769.pdf.

[21] National Aeronautics and Space Administration page last updated Aug 3 2017 Page Editor Monroe Conner Nasa official Brian Dumbar B200 king air image galery (https)://www.nasa.gov/centers/armstrong/multimedia/imagegallery/KingAir/index.html.

[22] Parreño José Alfredo para la carrera ingeniería electromecánica Latacunga / utc / 2012 128 páginas Diseño y construcción de un banco de pruebas de control neumático, con touch panel y s7-1200 de la universidad técnica de Cotopaxi.

[23] Proyecto Texas Instrument El microcontrolador!MSP430g2553 Copyright © 2018, Texas Instruments Incorporated.

[24]PROMACCválvulaelectroneumáticaenlínea.http://www.promaccltda.com.co/product/valvula-electroneumática/ recuperado el 20 de mayo.

[25] ROME, co industrias, equipo industrial, neumática, alto inventario, en línea. ubicación: https://www.romecoindustrial.com/, Recuperado el 18 de mayo del 2020.

[26] Super king air 200/B200 Pilot Training Manual FlightSafety International, Inc. Marine

Air Terminal, LaGuardia Airport Flushing, New York 11371 (718) 565-4100 (https)://www.flightsafety.com/.

[27] Tutorial processing Clase automatización con arduino Automatización de sistemas Industriales en la Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Azcapotzalco Copyright © 2020.

[28]Texas Instrument MSP430G2553 LaunchPad™ Development Kit (MSP-EXP430G2ET) User's Guide SLAU772–June 2018 Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265 Copyright © 2018, Texas Instruments Incorporated.

[29]Tutorialprocessing<https://arquiblog.uniandes.edu.co/blogs/arqu3503/files/2009/02/processing_workshop.pdf>.

[30]TarjetaArudirnomega2560-https://www.electrontools.com/Home/WP/arduino-mega-2560-caracteristicas/.

[31]TutorialProcessing3.0.https://contenedor-digital.buenosaires.gob.ar/descargar/760aac-tutorial-processing-3.pdf.

[32] Unidad administrativa especial de aeronáutica civil RAC 43 http://www.aerocivil.gov.co/normatividad/RAC/RAC%20%2043%20-%20Mantenimiento.pdf.

[33] VERCOIL HIDRÁULICA Válvula proporcional de caudal en línea https://www.vercoil.com/shop/bosch-rexroth/valvula-proporcional-caudal-fes recuperado el 19 de mayo del 2020.

[34] WORDPRESS, Aprendiendo arduino - Sensores y Actuadores, en línea. Ubicación: https://aprendiendoarduino.wordpress.com/tag/actuadores/. Recuperado el 18 de mayo del 2020.

Apéndice A

Sintaxis de programación en Arduino y Energía

- **Estructura básica del lenguaje de programación.**

Funciones. Es un bloque de código que tiene un nombre y un conjunto de estamentos que son ejecutados cuando se llama la función.

Condiciones

. Es una sentencia la cual se puede determinar si es falsa o es verdadera

Estamentos: Son el conjuntos de instrucciones o declaraciones que van contenidas dentro de los bloques de código.

Estructura básica: La estructura básica es bastante simple y está compuesta por dos partes fundamentales para la funcionalidad.

```
void setup()
{
estamentos;
}
void loop()
{
estamentos;

}
```

Setup(): Contiene la configuración del programa, la declaración de variables, es la primera función del código, se coloca una sola vez, allí se inicia los modos de trabajo de los pin, se incluye así no hayan variables.

Loop(): Contiene el programa que se ejecuta cíclicamente (bucle), es decir el código cíclico, donde se programan lecturas de entradas, activación de los puertos etc.

Entre llaves "{}": Indican el principio y el fin de todas las instrucciones insertadas en un bloque de programación, un ejemplo de cómo se aplican se evidencia de la siguiente forma:

Punto y coma ";": separa las instrucciones y elementos de la programación y también las instrucciones en un bucle, por ejemplo un bucle de tipo for.

Bloques de comentarios: No se tienen en cuenta al momento de correr el programa, solo sirven para describir pasos e información de los procedimientos del programa, estos se

pueden usar de la siguiente forma:

```
/* esto es un bloque de comentario
no se debe olvidar cerrar los comentarios
 estos deben estar equilibrados */
```

Línea de comentarios: Cumple una funcion similar al bloque de comentarios, la diferencia es que está solo es para una línea.

```
// esto es un comentario.
```

Declaración de variables

Todas las variables tienen que declararse antes de que puedan ser utilizadas para declarar una variable se comienza por definir su tipo como **int** (entero),**long** (largo) **float** (coma flotante), etc, asignándoles siempre un nombre y ,opcionalmente, un valor inicial. Esto solo debe hacerse una vez, pero el valor se puede cambiar en cualquier momento usando aritmética y reasignación diversas.

Byte

Byte almacena un valor numérico de 8 bits sin decimales. Tienen un rango entre 0 y 255

Int

Enteros son un tipo de datos primarios que almacenan valores numéricos de 16 bits sin decimales comprendidos en el rango 32,767 a -32,768.

Long

El formato de variable numérica de tipo extendido "long" se refiere a números enteros (tipo 32 bits) sin decimales que se encuentran dentro del rango -2147483648 a 2147483647

Float

El dato del tipo "punto flotante" "float" se aplica a los números decimales. Los números de punto flotante tienen una mayor resolución que los de 32 bits con un rango comprendido *3.4028235E +38 a +38-3.4028235E.*

Aritmética

Los operadores aritméticos que se incluyen en el entorno de programación son suma, resta, multiplicación y división. Estos devuelven la suma, diferencia, producto, o cociente (respectivamente) de dos operandos.

```
y = y + 3;
x = x - 7;
i = j * 6;
r = r / 5;
```

La operaciones se efectúa teniendo en cuenta el tipo de datos que hemos definido para los operandos (int, dbl, float, etc..), por lo que, por ejemplo, si definimos 9 y 4 como enteros

“int”, 9 / 4 devuelve de resultado 2 en lugar de 2,25 ya que el 9 y 4 se valores de tipo entero “int” (enteros) y no se reconocen los decimales con este tipo de datos.

Esto también significa que la operación puede sufrir un desbordamiento si el resultado es más grande que lo que puede ser almacenada en el tipo de datos. Recordemos el alcance de los tipos de datos numéricos que ya hemos explicado anteriormente.

Si los operandos son de diferentes tipos, para el cálculo se utilizará el tipo más grande de los operandos en juego. Por ejemplo, si uno de los números (operandos) es del *tipo float* y otra de *tipo integer*, para el cálculo se utilizará el *método de float* es decir el método de coma flotante.

Elija el tamaño de las variables de tal manera que sea lo suficientemente grande como para que los resultados sean lo precisos que usted desea. Para las operaciones que requieran decimales utilice variables *tipo float*, pero sea consciente de que las operaciones con este tipo de variables son más lentas a la hora de realizarse el cómputo.

Utilizando el operador **(int) *myFloat*** para convertir un tipo de variable a otro sobre la marcha. Por ejemplo, **i = (int) 3,6** establecerá **i** igual a **3.**

Operadores de Comparación

Las comparaciones de una variable o constante con otra se utilizan con frecuencia en las estructuras condicionales del tipo *if..* para testear si una condición es verdadera. En los ejemplos que siguen en las próximas páginas se verá su utilización práctica usando los siguientes tipo de condicionales:

```
x == y   //   x es igual a y
x != y   //   x no es igual a y
x < y    //   x es menor que y
x > y    //   x es mayor que y
x <= y   //   x es menor o igual que y
x >= y   //   x es mayor o igual que y
```

Operadores lógicos

Los operadores lógicos son usualmente una forma de comparar dos expresiones y devolver un **VERDADERO o FALSO** dependiendo del operador. Existen tres operadores lógicos, **AND (&&), OR (||) y NOT (!),** que a menudo se utilizan en estamentos de tipo if..:

```
Logical AND:
if (x > 0 && x < 5)      // cierto sólo si las dos expresiones son ciertas
Logical OR:
if (x > 0 || y > 0)      // cierto si una cualquiera de las expresiones es cierta
Logical NOT:
if (!x > 0)              // cierto solo si la expresión es falsa
```

Constantes

El lenguaje de programación de Arduino tiene unos valores predeterminados, que son llamados constantes. Se utilizan para hacer los programas más fáciles de leer. Las constantes se

clasifican en grupos.

Cierto/falso(true/false)

Estas son constantes booleanas que definen los niveles **HIGH** (alto) y **LOW** (bajo) cuando estos se refieren al estado de las salidas digitales. **FALSE** se asocia con 0 (cero), mientras que TRUE se asocia con 1, pero **TRUE** también puede ser cualquier otra cosa excepto cero. Por lo tanto, en sentido booleano, -1, 2 y -200 son todos también se define como **TRUE**.

```
if (b == TRUE);
{
ejecutar las instrucciones;
}
```

High/Low

Estas constantes definen los niveles de salida altos o bajos y se utilizan para la lectura o la escritura digital para las patillas. A**LTO** se define como en la lógica de nivel 1, **ON**, ó 5 voltios, mientras que **BAJO** es lógica nivel 0, **OFF**, o 0 voltios.

```
digitalWrite(13, HIGH); // activa la salida 13 con un nivel alto (5v.)
```

Input/Output

Estas constantes son utilizadas para definir, al comienzo del programa, el modo de funcionamiento de los pines mediante la instrucción pinMode de tal manera que el pin puede ser una **entrada INPUT** o una **salida OUTPUT.**

```
pinMode(13, OUTPUT); // designamos que el PIN 13 es una salida
```

If/(si)

if es un estamento que se utiliza para probar si una determinada condición se ha alcanzado, como por ejemplo averiguar si un valor analógico está por encima de un cierto número, y ejecutar una serie de declaraciones (operaciones) que se escriben dentro de llaves, si es verdad. Si es falso (la condición no se cumple) el programa salta y no ejecuta las operaciones que están dentro de las llaves, El formato para if es el siguiente:

```
if (unaVariable ?? valor)
{
ejecutaInstrucciones;
 }
```

Se compara una variable con un valor, el cual puede ser una variable o constante. Si la comparación, o la condición entre paréntesis se cumple (es cierta), las declaraciones dentro de los corchetes se ejecutan. Si no es así, el programa salta sobre ellas y sigue.

El uso especial del símbolo '=', poner dentro de if (x = 10), podría parecer que es válido pero sin embargo no lo es ya que esa expresión asigna el valor 10 a la variable x, por eso dentro de la estructura if se utilizaría X==10 que en este caso lo que hace el programa es comprobar si el valor de x es 10.. Ambas cosas son distintas por lo tanto dentro de las estructuras if, cuando se pregunte por un valor se debe poner el signo doble de igual "=="

If...else (si.....sino..)

if... else viene a ser una estructura que se ejecuta en respuesta a la idea *"si esto no se cumple haz esto otro"*. Por ejemplo, si se desea probar una entrada digital, y hacer una cosa si la entrada fue alto o hacer otra cosa si la entrada es baja, usted escribiría que de esta manera:

```
if (inputPin == HIGH) //si el valor de la entrada inputPin es alto
      {
            instruccionesA; //ejecuta si se cumple la condición
       }
else
      {
             instruccionesB; //ejecuta si no se cumple la condición
      }
```

Else puede ir precedido de otra condición de manera que se pueden establecer varias estructuras condicionales de tipo unas dentro de las otras (anidamiento) de forma que sean mutuamente excluyentes pudiéndose ejecutar a la vez. Es incluso posible tener un número ilimitado de estos condicionales. Recuerde sin embargo que sólo un conjunto de declaraciones se llevará a cabo dependiendo de la condición probada:

```
if (inputPin < 500)
     {
          instruccionesA; // ejecuta las operaciones A
     }
else if (inputPin >= 1000)
     {
          instruccionesB; // ejecuta las operacione B
     }
else
     {
          instruccionesC; // ejecuta las operaciones C
     }
```

Un estamento de tipo *if* prueba simplemente si la condición dentro del paréntesis es verdadera o falsa. Esta declaración puede ser cualquier declaración válida. En el anterior ejemplo, si cambiamos y ponemos (inputPin == HIGH). En este caso, el estamento if sólo verificará si la entrada especificado esta en nivel alto (HIGH), o +5 v.

PinMode(pin , mode)

Esta instrucción es utilizada en la parte de configuración setup () y sirve para configurar el modo de trabajo de un **PIN** pudiendo ser INPUT (entrada) y **OUTPUT** (salida).

pinMode(pin, OUTPUT); *// configura 'pin' como salida*

Los terminales de Arduino, por defecto, están configurados como entradas, por lo tanto no es necesario definirlos en el caso de que vayan a trabajar como entradas. Los pines configurados como entrada quedan, bajo el punto de vista eléctrico, como entradas en estado de alta impedancia.

Estos pines tienen a nivel interno una resistencia de 20 KΩ a las que se puede acceder mediante software. Estas resistencias se accede de la siguiente manera:

pinMode(pin, INPUT); *// configura el 'pin' como entrada*

```
digitalWrite(pin, HIGH); // activa las resistencias internas
```

Las resistencias internas normalmente se utilizan para conectar las entradas a interruptores. En el ejemplo anterior no se trata de convertir un pin en salida, es simplemente un método para activar las resistencias interiores

Los pines configurados como OUTPUT (salida) se dice que están en un estado de baja impedancia estado y pueden proporcionar **40 mA** (miliamperios) de corriente a otros dispositivos y circuitos. Esta corriente es suficiente para alimentar un diodo LED (no olvidando poner una resistencia en serie), pero no es lo suficientemente grande como para alimentar cargas de mayor consumo como relés, solenoides, o motores.

Un cortocircuito en las patillas Arduino provocará una corriente elevada que puede dañar o destruir el chip Atmega. A menudo es una buena idea conectar en la OUTUPT (salida) una resistencia externa de 470 o de 1000 Ω.

DigitalRead(pin)

Lee el valor de un pin (definido como digital) dando un resultado HIGH (alto) o LOW (bajo). El pin se puede especificar ya sea como una variable o una constante (0-13).

```
valor = digitalRead(Pin); // hace que 'valor sea igual al estado leído en 'Pin'
```

DigitalWrite(pin, value)

Envía al ´pin´ definido previamente como OUTPUT el valor HIGH o LOW (poniendo en 1 o 0 la salida). El pin se puede especificar ya sea como una variable o como una constante (0-13).

```
digitalWrite(pin, HIGH); // deposita en el 'pin' un valor HIGH (alto o 1)
```

El siguiente ejemplo lee el estado de un pulsador conectado a una entrada digital y lo escribe en el ´pin´de salida LED:

```
int led = 13; // asigna a LED el valor 13
int boton = 7; // asigna a botón el valor 7
 int valor = 0; // define el valor y le asigna el valor 0
void setup()
{
pinMode(led, OUTPUT); // configura el led (pin13) como salida
pinMode(boton, INPUT); // configura botón (pin7) como entrada
}
void loop()
{
 valor = digitalRead(boton); //lee el estado de la entrada botón
digitalWrite(led, valor); // envía a la salida 'led'el valor leído
 }
```

AnalogRead(pin)

Lee el valor de un determinado pin definido como entrada analógica con una resolución de 10 bits. Esta instrucción sólo funciona en los pines (0-5). El rango de valor que podemos leer

oscila de 0 a 1023.

```
valor = analogRead(pin); // asigna a valor lo que lee en la entrada 'pin'
```

Los pins analógicos (0-5) a diferencia de los pines digitales, no necesitan ser declarados como INPUT u OUPUT ya que son siempre INPUT´s.

AnalogWrite(pin, value)

Esta instrucción sirve para escribir un pseudo-valor analógico utilizando el procedimiento de modulación por ancho de pulso (PWM) a uno de los pin´s de Arduino marcados como "pin PWM". El más reciente Arduino, que implementa el chip **ATmega168, permite habilitar como salidas analógicas tipo PWM los pines 3, 5, 6, 9, 10 y 11.** Los modelos de Arduino más antiguos que implementan el chip ATmega8, solo tiene habilitadas para esta función los pines 9, 10 y 11. El valor que se puede enviar a estos pines de salida analógica puede darse en forma de variable o constante, pero siempre con un margen de 0-255.

analogWrite(pin, valor); *// escribe 'valor' en el 'pin' definido como analógico*

Si enviamos el valor 0 genera una salida de 0 voltios en el pin especificado; un valor de 255 genera una salida de 5 voltios de salida en el pin especificado. Para valores de entre 0 y 255, el pin saca tensiones entre 0 y 5 voltios - el valor HIGH de salida equivale a 5v (5 voltios). Teniendo en cuenta el concepto de señal PWM , por ejemplo, un valor de 64 equivaldrá a mantener 0 voltios de tres cuartas partes del tiempo y 5 voltios a una cuarta parte del tiempo; un valor de 128 equivaldrá a mantener la salida en 0 la mitad del tiempo y 5 voltios la otra mitad del tiempo, y un valor de 192 equivaldrá a mantener en la salida 0 voltios una cuarta parte del tiempo y de 5 voltios de tres cuartas partes del tiempo restante.

Debido a que esta es una función de hardware, en el pin de salida analógica (PWN) se generará una onda constante después de ejecutada la instrucción analogWrite hasta que se llegue a ejecutar otra instrucción analogWrite (o una llamada a digitalRead o digitalWrite en el mismo pin).

Las salidas analógicas a diferencia de las digitales, no necesitan ser declaradas como INPUT u OUTPUT.. El siguiente ejemplo lee un valor analógico de un pin de entrada analógica, convierte el valor dividiéndolo por 4, y envía el nuevo valor convertido a una salida del tipo PWM o salida analógica:

```
int led = 10; // define el pin 10 como 'led'
int analog = 0; // define el pin 0 como 'analog'
int valor; // define la variable 'valor'
void setup(){} // no es necesario configurar entradas y salidas
void loop()
{
     valor = analogRead(analog); // lee el pin 0 y lo asocia a la variable valor
     valor /= 4; //divide valor entre 4 y lo reasigna a valor
    analogWrite(led, value); // escribe en el pin10 valor
}
```

Delay(ms)

Detiene la ejecución del programa la cantidad de tiempo que indica la propia instrucción. De

tal manera que 1000 equivale a 1 seg.

```
delay(1000); // espera 1 segundo
```

Serial.begin(rate)

Abre el puerto serie y fija la velocidad en baudios para la transmisión de datos en serie. El valor típico de velocidad para comunicarse con el ordenador es 9600, aunque otras velocidades pueden ser soportadas.

```
void setup()
{
Serial.begin(9600); // abre el Puerto serie
 } // configurando la velocidad en 9600 bps
```

Cuando se utiliza la comunicación serie los pins digital 0 *(RX)* y 1 *(TX)* no puede utilizarse al mismo tiempo.

Serial.println(data)

Imprime los datos en el puerto serie, seguido por un retorno de carro automático y salto de línea. Este comando toma la misma forma que Serial.print (), pero es más fácil para la lectura de los datos en el Monitor Serie del software.

```
Serial.println(analogValue); // envía el valor 'analogValue' al puerto
```

Para obtener más información sobre las distintas posibilidades de Serial.println () y Serial.print () puede consultarse el sitio web de Arduino. El siguiente ejemplo toma de una lectura analógica pin0 y envía estos datos al ordenador cada 1 segundo.

```
void setup()
    {
        Serial.begin(9600); // configura el puerto serie a 9600bps
    }
 void loop()
    {
        Serial.println(analogRead(0)); // envía valor analógico
        delay(1000);                   // espera 1 segundo
}
```

Serial.printnl(data, data type)

Vuelca o envía un número o una cadena de caracteres al puerto serie, seguido de un carácter de retorno de carro "CR" (ASCII 13, or '\r')y un carácter de salto de línea "LF"(ASCII 10, or '\n'). Toma la misma forma que el comando Serial.print()

Serial.println(b) vuelca o envía el valor de b como un número decimal en caracteres ASCII seguido de "CR" y "LF".

Serial.println(b, DEC) vuelca o envía el valor de b como un número decimal en caracteres ASCII seguido de "CR" y "LF".

Serial.println(b, HEX) vuelca o envía el valor de b como un número hexadecimal en caracteres ASCII seguido de "CR" y "LF".

Serial.println(b, OCT) vuelca o envía el valor de b como un número Octal en caracteres ASCII seguido de "CR" y "LF".

Serial.println(b, BIN) vuelca o envía el valor de b como un número binario en caracteres ASCII seguido de "CR" y "LF".

Serial.print(b, BYTE) vuelca o envía el valor de b como un byteseguido de "CR" y "LF".

Serial.println(str) vuelca o envía la cadena de caracteres como una cadena ASCII seguido de "CR" y "LF".

Serial.println() sólo vuelca o envía "CR" y "LF". Equivaldría a printNewline()**.**

Serial.print(data, data type)

Vuelca o envía un número o una cadena de caracteres, al puerto serie. Dicho comando puede tomar diferentes formas, dependiendo de los parámetros que utilicemos para definir el formato de volcado de los números.

Parámetros data: el número o la cadena de caracteres a volcar o enviar.

data type: determina el formato de salida de los valores numéricos (decimal, octal, binario, etc...) DEC, OCT, BIN, HEX, BYTE , si no se puede ,vuelva ASCII

Ejemplos:

Serial.print(b)

Vuelca o envía el valor de b como un número decimal en caracteres ASCII. Equivaldría a printInteger().

```
int b = 79; Serial.print(b); // prints the string "79".
```

Serial.print(b, DEC)

Vuelca o envía el valor de b como un número decimal en caracteres ASCII. Equivaldría a printInteger().

```
int b = 79;
Serial.print(b, DEC); // prints the string "79".
Serial.print(b, HEX)
```

Vuelca o envía el valor de b como un número hexadecimal en caracteres ASCII. Equivaldría a printHex();

```
int b = 79;
 Serial.print(b, HEX); // prints the string "4F".
Serial.print(b, OCT)
```

Vuelca o envía el valor de b como un número Octal en caracteres ASCII. Equivaldría a printOctal();

```
int b = 79;
Serial.print(b, OCT); // prints the string "117".
Serial.print(b, BIN)
```

Vuelca o envía el valor de b como un número binario en caracteres ASCII. Equivaldría a printBinary();

```
int b = 79;
Serial.print(b, BIN); // prints the string "1001111".
Serial.print(b, BYTE)
```

Vuelca o envía el valor de b como un byte. Equivaldría a printByte();

```
int b = 79;
Serial.print(b, BYTE); // Devuelve el carácter "O", el cual representa el carácter ASCII del valor 79. (Ver tabla ASCII ).
Serial.print(str)
```

Vuelca o envía la cadena de caracteres como una cadena ASCII. Equivaldría a printString().

```
Serial.print("Hello World!"); // vuelca "Hello World!"
```

Serial.available()

int Serial.available()

Obtiene un número entero con el número de bytes (caracteres) disponibles para leer o capturar desde el puerto serie. Equivaldría a la función serialAvailable().

Devuelve un entero con el número de bytes disponibles para leer desde el buffer serie, o 0 si no hay ninguno. Si hay algún dato disponible, SerialAvailable() será mayor que 0. El buffer serie puede almacenar como máximo 64 bytes.

Ejemplo:

```
int incomingByte = 0; // almacena el dato serie
void setup()
{
Serial.begin(9600); // abre el puerto serie, y le asigna la velocidad de 9600 bps
}
void loop() {
      // envía datos sólo si los recibe:
     if (Serial.available() > 0) {
                 // lee el byte de entrada:
               incomingByte = Serial.read();
            //lo vuelca a pantalla
                Serial.print("I received: ");
                Serial.println(incomingByte, DEC);
       }
}
```

Serial.Read()

int Serial.Read()

Lee o captura un byte (un carácter) desde el puerto serie. Equivaldría a la función serialRead().

Devuelve :El siguiente byte (carácter) desde el puerto serie, o -1 si no hay ninguno.

Ejemplo:

```
int incomingByte = 0; // almacenar el dato serie
void setup() {
Serial.begin(9600); // abre el puerto serie,y le asigna la velocidad de 9600 bps
}
void loop() {
    // envía datos sólo si los recibe:
   if (Serial.available() > 0) {
             // lee el byte de entrada:
             incomingByte = Serial.read();
           //lo vuelca a pantalla
```

```
                Serial.print("I received: ");
                Serial.println(incomingByte, DEC);
                }
}
```

Apéndice B

Sintaxis de programación Processing

Variables Processing

Name Import

Ejemplo:

```
import processing.pdf.*;
 void setup()
 {
size(1024, 768, PDF);
  }
void draw() {
 line(0, 0, width, height);
```

Descripción

La importación de palabras clave se utiliza para cargar una biblioteca en un boceto de procesamiento. Una biblioteca es una o más clases que se agrupan para ampliar las capacidades de procesamiento. El carácter * se utiliza a menudo al final de la línea de importación (consulte el ejemplo de código anterior) para cargar todas las clases relacionadas a la vez, sin tener que hacer referencia a ellas individualmente la declaración de importación se creará seleccionando una biblioteca en "Importar

Name boolean

Ejemplo:

```
boolean a = false;
if (!a) {
 rect(30, 20, 50, 50);
 }
 a = true;
 if (a) {
 line(20, 10, 90, 80);
 line(20, 80, 90, 10);
 }
```

Descripción

Tipo de datos para los valores booleanos verdadero y falso. Es común usar valores booleanos con declaraciones de control para determinar el flujo de un programa. La primera vez que se escribe una variable, debe declararse con una declaración que exprese su tipo de datos.

Name false

Ejemplo:

```
 rect(30, 20, 50, 50)
boolean b = false;
 if (b == false) {
line(20, 10, 90, 80);  // This line is drawn
 } else {
 line(20, 80, 90, 10);  // This line is not drawn
                }
```

Descripción

Palabra reservada que representa el valor lógico "falso". Solo a las variables de tipo booleano se les puede asignar el valor falso.

Name String

Ejemplo:

```
String str1 = "CCCP";
char data[] = {'C', 'C', 'C', 'P'};
String str2 = new String(data);
println(str1);  // Prints "CCCP" to the console
println(str2);  // Prints "CCCP" to the console
 // Comparing String objects, see reference below
 String p = "potato";
 // The correct way to compare two Strings
  if (p.equals("potato")) {
 println("Yes, the values are the same.");
 }
 // Use a backslash to include quotes in a String
 String quoted = "This one has \"quotes\"";
  println(quoted);  // This one has "quotes"
```

Descripción

Una cadena es una secuencia de caracteres. La clase String incluye métodos para examinar caracteres individuales, comparar cadenas, buscar cadenas, extraer partes de cadenas y convertir una cadena completa en mayúsculas y minúsculas. Las cadenas siempre se definen entre comillas dobles ("Abc") y los caracteres siempre se definen entre comillas simples ('A'). Para comparar el contenido de dos cadenas, use el método equals (), como en if (a.equals (b)), en lugar de if (a == b). Una cadena es un objeto, por lo que compararlos con el operador == solo compara si ambas cadenas están almacenadas en la misma ubicación de memoria. El uso del método equals () asegurará que se comparen los contenidos reales. (La referencia de solución de problemas tiene una explicación más extensa). Debido a que una Cadena se define entre comillas dobles, para incluir dichas marcas dentro de la Cadena debe usar el carácter \ (barra invertida). (Consulte el tercer ejemplo anterior). Esto se conoce como secuencia de escape. Otras secuencias de escape incluyen \ t para el carácter de tabulación y \ n para la nueva línea. Porque la barra invertida es el escape. carácter, para incluir una sola barra invertida dentro de una cadena, debe usar dos barras invertidas consecutivas, como en: \\ Hay más métodos de cadena que los vinculados desde esta página. La documentación

adicional se encuentra en línea en la documentación oficial de Java.

Name Void

Ejemplo:

```
 void setup() {  // setup() does not return a value
size(200, 200);
 }
 void draw() {  // draw() does not return a value
  line(10, 100, 190, 100);
  drawCircle();
   }
   void drawCircle() {  // This function also does not return a value
 ellipse(30, 30, 50, 50);
 }
```

Descripción

La palabra clave utilizada indica que una función no devuelve ningún valor. Cada función debe devolver un valor de un tipo de datos específico o utilizar la palabra clave void para especificar que no devuelve nada.

size(width, height)

size(width, height, renderer)

Name new

Ejemplos:

```
HLine h1 = new HLine();
float[] speeds = new float[3];
float ypos;
void setup() {
size(200, 200);
speeds[0] = 0.1;
speeds[1] = 2.0;
speeds[2] = 0.5
}
void draw() {
ypos += speeds[int(random(3))];
if (ypos > width) {
ypos = 0;
}
h1.update(ypos);
}
class HLine {
void update(float y) {
line(0, y, width, y);
}
}
```

Descripción

Crea un objeto "nuevo". La palabra clave nueva se utiliza normalmente de forma similar a las aplicaciones del ejemplo anterior. En este ejemplo, se crea un nuevo objeto "h1" del tipo de datos "HLine". En la siguiente línea, se crea una nueva matriz de flotadores

denominada "velocidades".

Name this

Ejemplos:

```
float ypos = 50;
void setup() {
size(100, 100);
noLoop();
}
void draw() {
line(0, 0, 100, ypos);
// "this" references the Processing sketch,
// and is not necessary in this case
this.ypos = 100;
line(0, 0, 100, ypos);
}
import processing.video.*;
Movie myMovie;

void setup() {
size(200, 200);
background(0);
// "this" references the Processing sketch
myMovie = new Movie(this, "totoro.mov");
myMovie.loop();
}
void draw() {
if (myMovie.available()) {
myMovie.read();
}
image(myMovie, 0, 0);
}
```

Descripción

Se refiere al objeto actual (es decir, "este objeto"), que cambiará según el contexto al que se haga referencia. En Processing, es más común usar esto para pasar una referencia del objeto actual a una de las bibliotecas.

Name draw()

Ejemplos:

```
float yPos = 0.0;
void setup() {  // setup() runs once
 size(200, 200);
frameRate(30);
}
void draw() {  // draw() loops forever, until stopped
 background(204);
 yPos = yPos - 1.0;
 if (yPos < 0) {
 yPos = height;
 }
 line(0, yPos, width, yPos);
}
void setup() {
size(200, 200);
}
```

```
// Although empty here, draw() is needed so
// the sketch can process user input events
// (mouse presses in this case).
void draw() { }
void mousePressed() {
 line(mouseX, 10, mouseX, 90);
}
```

Descripción

Llamada directamente después de setup (), la función draw () ejecuta continuamente las líneas de código contenidas dentro de su bloque hasta que se detiene el programa o se llama a noLoop (). draw () se llama automáticamente y nunca debería llamarse explícitamente. Todos los programas de procesamiento actualizan la pantalla al final del sorteo (), nunca antes.

Name background()

Examples

```
background (255, 204, 0);
background(51)
Syntax
background(rgb)
background(rgb, alpha)
background(gray)
background(gray, alpha)
background(v1, v2, v3)
background(v1, v2, v3, alpha)
background(image)
```

Descripción

La función background () establece el color utilizado para el fondo de la ventana Procesando. El fondo predeterminado es gris claro. Esta función se usa generalmente dentro de draw () para borrar la ventana de visualización al comienzo de cada fotograma, pero se puede usar dentro de setup () para establecer el fondo en el primer fotograma de la animación o si el fondo solo necesita establecerse una vez

Name fill()

Ejemplos

```
 fill(153);
rect(30, 20, 55, 55)
fill(204, 102, 0);
rect(30, 20, 55, 55);
```

Descripción

Establece el color utilizado para rellenar formas. Por ejemplo, si ejecuta fill (204, 102, 0), todas las formas posteriores se rellenará con naranja. Este color se especifica en términos de color RGB o HSB según el colorMode () actual. El espacio de color predeterminado es RGB, con cada valor en el rango de 0 a 255.

Name **rect()**

Ejemplos:

```
rect(30, 20, 55, 55);
rect(30, 20, 55, 55, 7);
rect(30, 20, 55, 55, 3, 6, 12, 18)
```

Descripción

Dibuja un rectángulo en la pantalla. Un rectángulo es una forma de cuatro lados con cada ángulo de noventa grados. De forma predeterminada, los dos primeros parámetros establecen la ubicación de la esquina superior izquierda, el tercero establece el ancho y el cuarto establece la altura. Sin embargo, la forma en que se interpretan estos parámetros se puede cambiar con la función rectMode ().

Name **setup()**

Examples

```
int x = 0;
void setup() {
size(200, 200);
background(0);
noStroke();
fill(102);
}
void draw() {
rect(x, 10, 2, 80);
x = x + 1;
}
int x = 0;
void setup() {
fullScreen();
background(0);
noStroke();
 fill(102);
}
void draw() {
 rect(x, height*0.2, 1, height*0.6);
 x = x + 2;
}
```

Descripción

La función setup () se ejecuta una vez, cuando se inicia el programa. Se utiliza para definir las propiedades del entorno inicial, como el tamaño de la pantalla, y para cargar medios como imágenes y fuentes cuando se inicia el programa. Solo puede haber una función setup () para cada programa y no debería volver a llamarse después de su ejecución inicial

Name **ellipse()**

Ejemplo:

```
ellipse(56, 46, 55, 55)
```

Descripción:

Dibuja una elipse (óvalo) en la pantalla. Una elipse con el mismo ancho y alto es un círculo. De forma predeterminada, los dos primeros parámetros establecen la ubicación, y el tercer y cuarto parámetros establecen el ancho y alto de la forma. El origen se puede cambiar con la función ellipseMode ().

Name textAlign()

Examples

```
background(0);
textSize(16);
textAlign(RIGHT);
text("ABCD", 50, 30);
textAlign(CENTER);
text("EFGH", 50, 50);
textAlign(LEFT);
text("IJKL", 50, 70);
background(0);
stroke(153);
textSize(11);
textAlign(CENTER, BOTTOM);
line(0, 30, width, 30);
text("CENTER,BOTTOM", 50, 30);
textAlign(CENTER, CENTER);
line(0, 50, width, 50);
text("CENTER,CENTER", 50, 50);
textAlign(CENTER, TOP);
line(0, 70, width, 70);
text("CENTER,TOP", 50, 70);
```

Descripción

Establece la alineación actual para dibujar texto. Los parámetros IZQUIERDA, CENTRAL y DERECHA establecen las características de visualización de las letras en relación con los valores de los parámetros xey de la función text ().

Name text Size()

Ejemplos:

```
background(0);
fill(255);
textSize(26);
text("WORD", 10, 50);
textSize(14);
text("WORD", 10, 70);
```

Descripción

Establece el tamaño de fuente actual. Este tamaño se utilizará en todas las llamadas posteriores a la función text (). El tamaño de la fuente se mide en unidades de píxeles.

Name text()

text Size(size)

Ejemplos:

```
textSize(32);
text("word", 10, 30);
fill(0, 102, 153);
text("word", 10, 60);
fill(0, 102, 153, 51);
text("word", 10, 90);

size(100, 100, P3D);
textSize(32);
fill(0, 102, 153, 204);
text("word", 12, 45, -30);  // Specify a z-axis value
text("word", 12, 60);  // Default depth, no z-value specified
String s = "The quick brown fox jumps over the lazy dog.";
fill(50);
text(s, 10, 10, 70, 80);  // Text wraps within text box
```

Descripción

Dibuja texto en la pantalla. Muestra la información especificada en el primer parámetro en la pantalla en la posición especificada por los parámetros adicionales. Se usará una fuente predeterminada a menos que se configure una fuente con la función textFont () y se usará un tamaño predeterminado a menos que se configure una fuente con textSize (). Cambie el color del texto con la función fill (). El texto se muestra en relación con la función textAlign (), que da la opción de dibujar a la izquierda, a la derecha y al centro de las coordenadas

Syntax

```
text(c, x, y)
text(c, x, y, z)
text(str, x, y)
text(chars, start, stop, x, y)
text(str, x, y, z)
text(chars, start, stop, x, y, z)
text(str, x1, y1, x2, y2)
text(num, x, y)
text(num, x, y, z)

Name     if
for (int i = 5; i < height; i += 5) {
  stroke(255);   // Set the color to white
  if (i < 35) {  // When 'i' is less than 35...
    stroke(0);   //...set the color to black
  }
  line(30, i, 80, i);
}

  Syntax
if (test) {
  statements
}
```

Descripción: Permite al programa tomar una decisión sobre qué código ejecutar. Si la prueba se evalúa como verdadera, las declaraciones incluidas dentro del bloque se ejecutan y si la prueba se evalúa como falsa, las declaraciones no se ejecutan.

Name mouseClicked()

Examples:

```
// Click within the image to change
// the value of the rectangle after
// after the mouse has been clicked
int value = 0;
void draw() {
 fill(value);
 rect(25, 25, 50, 50);
}
void mouseClicked() {
 if (value == 0) {
  value = 255;
 } else {
 value = 0;
 }
}
```

Descripción

La función mouseClicked () se llama después de presionar y soltar un botón del mouse. Los eventos de mouse y teclado solo funcionan cuando un programa tiene draw (). Sin draw (), el código solo se ejecuta una vez y luego deja de escuchar eventos.

Name float

Examples

```
float a;        // Declare variable 'a' of type float
a = 1.5387;     // Assign 'a' the value 1.5387
float b = -2.984;  // Declare variable 'b' and assign it the value -2.984
float c = a + b;   // Declare variable 'c' and assign it the sum of 'a' and 'b'
float f1 = 0.0;
for (int i = 0 ; i < 100000; i++) {
 f1 = f1 + 0.0001;  // Bad idea! See below.
}
println(f1);

float f2 = 0.0;
for (int i = 0; i < 100000; i++) {
 // The variable 'f2' will work better here, less affected by rounding
 f2 = i / 1000.0;  // Count by thousandths
}
println(f2);
```

Descripción

Tipo de datos para números de coma flotante, p. Ej. Los números que tienen un punto

decimal los flotantes no son precisos, por lo que la adición de valores pequeños (como 0,0001) no siempre puede aumentar con precisión debido a errores de redondeo. Si desea incrementar un valor en intervalos pequeños, use un int y divida por un valor flotante antes de usarlo. (Vea el segundo ejemplo anterior).

float var

Ejemplos

```
float var = value
Name    mouseX
void draw() {
  background(204);
  line(mouseX, 20, mouseX, 80);
}
```

Descripción

La variable de sistema mouseX siempre contiene la coordenada horizontal actual del mouse, tenga en cuenta que Processing solo puede rastrear la posición del mouse cuando el puntero está sobre la ventana actual. El valor predeterminado de mouseX es 0, por lo que se devolverá 0 hasta que el mouse se mueva frente a la ventana del boceto. (Esto suele ocurrir cuando se ejecuta un boceto por primera vez). Una vez que el mouse se aleja de la ventana, mouseX continuará informando su posición más reciente.

Name mouseY

Ejemplos:

```
void draw() {
 background(204);
 line(20, mouseY, 80, mouseY);
}
```

Descripción

La variable de sistema mouse siempre contiene la coordenada vertical actual del mouse tenga en cuenta que Processing solo puede rastrear la posición del mouse cuando el puntero está sobre la ventana actual. El valor predeterminado de mouseY es 0, por lo que se devolverá 0 hasta que el mouse se mueva frente a la ventana del boceto. (Esto suele ocurrir cuando se ejecuta un boceto por primera vez). Una vez que el mouse se aleja de la ventana, mouseY continuará informando su posición más reciente.

Apéndice C

Código de programación de la tarjeta Arduino Mega 2560 para el equipo de apoyo en tierra.

```
float lectura;
float volt = 0;
float voltvalve;
int VoutConver;
int Vout = 8;                //----- Número de los pines de distribución electrónica.
int VoltValveRight = 9;
int VoltValveLeft = 10;
int VoltValveRelief = 11;
char letra;
bool runningA = false;
bool runningB = false;
bool runningC = false;
bool runningD = false;
bool runningE = false;
bool Status = false;
bool RightValve = false;
bool LeftValve = false;
bool RightLeftValve = false;
bool SwitchR = false;
bool SwitchL = false;
bool SwitchRL = false;
int p = 0;
int Cycle = 0;
int countA = 0;
int countB = 0;
int countC = 0;
int countD = 0;
////////ACTIVACIÓN Y DEFINICIÓN DEL LOS PINES///////
void setup()
{
 Serial.begin(9600);
 pinMode(A0, INPUT);          // ----Activación del pin A0 para la entrada de voltaje del conversor del sensor.
 pinMode(VoltValveRight, OUTPUT);
 pinMode(VoltValveLeft, OUTPUT);
 pinMode(VoltValveRelief, OUTPUT);
}
void loop()
{
 lectura = analogRead(A0);
 volt = lectura / 1023 * 5;                    // Voltaje que entra a la tarjeta por el pin análogo.
 voltvalve = ( p * 25 ) / ( volt * 145 );          //Voltaje de salida para el conversor de la válvula proporcional..
 VoutConver = (voltvalve * 255) / 5;
 analogWrite(Vout, VoutConver);
 if (Serial.available() > 0)
 {
  letra = Serial.read();
  lectura = analogRead(A0);
  volt = lectura / 1023 * 5.0;
  if ((!runningE && letra == 'e') || runningE)
  {
   switch ( letra )
   {
    //////////////INSTRUCCIONES PARA EL  ESCENARIO A////////////
    case 'a':
```

```
countA = countA + 1;
if (!Status && countA == 1)
{
 Status = true;
 digitalWrite(VoltValveRelief, LOW);
 Serial.println(" Checking pressure...");
 delay(3000);

 if ( volt > 4.13 && volt < 4.82 )      // -----  Intervalo de voltaje para aceptar la presión de operación. para entrada de 0 - 5 v
 {
  Serial.println("Accepted pressure");
  delay(3000);
  digitalWrite(VoltValveRelief, HIGH);
  menu2();
 }
 else
 {
  Serial.println("Pressure not accepted");
  delay(3000);
  Serial.println("Restoring operation...");
  delay(1000);
  menu();
  Status = false;
 }
}
else if (countA == 1 && Status && !runningA && countB == 0 && countC == 0)
{
 runningA = true;
 runningB = false;
 runningC = false;
 runningD = false;
 Question1(90);
}
else if (Status && countA == 2 && runningA)
{
 LoadedPressure(90);
}
else if (Status && countA == 1 && countB == 1 && runningB)
{
 LoadedPressure(100);
}
else if (Status && countA == 1 && countC == 1 && runningC)
{
 LoadedPressure(120);
}
else if (Status && countA == 3 && countB == 0 && countC == 0)
{
 Question2(" right valve.");

 RightValve = true;
 runningD = false;
}
else if (Status && countA == 4 && countB == 0 && countC == 0 && RightValve)
{
 Serial.println("Opening right valve...");
 delay(3000);
 digitalWrite(VoltValveRight, HIGH);
 Confirmation2(" Right valve ");
 menu4(" Right valve ");
}
else if (Status && countA == 3 && countB == 1 && countC == 0 && LeftValve)
{
 Serial.println("Opening left valve...");
 delay(3000);
 digitalWrite(VoltValveLeft, HIGH);
 Confirmation2(" Left valve ");
 menu4(" Left valve ");
}
```

```
else if (Status && countA == 3 && countB == 0 && countC == 1 && RightLeftValve)
{
 Serial.println("Opening right and left valves...");
 delay(3000);
 digitalWrite(VoltValveLeft, HIGH);
 digitalWrite(VoltValveRight, HIGH);
 Confirmation2(" Right and left valves ");
 menu4(" Right and left valve ");
}
else if (Status && countA == 5 && countB == 0 && countC == 0 && RightValve)
{
 Question3();
 SwitchR = true;
}
else if (Status && countA == 6 && countB == 0 && countC == 0 && RightValve)
{
 Serial.println("Closing right valve...");
 delay(3000);
 Start Valves();
 Serial.println("Right valve closed");
 delay(3000);
 RightValve = false;
 Start Variables 2();
 menu3();
}
else if (Status && countA == 5 && countB == 1 && countC == 0 && RightValve && !SwitchR)
{
 Serial.println("Restoring operation...");
 delay(3000);
 menú();
}
else if (Status && countA == 4 && countB == 1 && countC == 0 && LeftValve)
{
 Question3();
 SwitchL = true;
}
else if (Status && countA == 5 && countB == 1 && countC == 0 && LeftValve)
{
 Serial.println("Closing left valve...");
 delay(3000);
 Start Valves();
 Serial.println("Left valve closed");
 delay(3000);
 Left Valve = false;
 Start Variables 2();
 menu3();
}
else if (Status && countA == 4 && countB == 2 && countC == 0 && LeftValve && !SwitchL)
{
 Serial.println("Restoring operation...");
 delay(1500);
 menú();
}
else if (Status && countA == 4 && countB == 0 && countC == 1 && RightLeftValve)
{
 Question3();
 SwitchRL = true;
}
else if (Status && countA == 5 && countB == 0 && countC == 1 && RightLeftValve)
{
 Serial.println("Closing right and left valves...");
 delay(3000);
 Start Valves();
 Serial.println("Right and left valves closed");
 delay(3000);
 RightLeftValve = false;
 Start Variables 2();
 menu3();
```

```
 }
 else if (Status && countA == 4 && countB == 1 && countC == 1 && RightLeftValve && !SwitchRL)
 {
  Serial.println("Restoring operation...");
  delay(3000);
  menú();
 }
 else
 {
  countA = countA - 1;
 }
  break;
 //////////////INSTRUCCIONES PARA EL EL ESCENARIO B/////////////
case 'b':
 //Serial.println(" Entro en el case b ");
 countB = countB + 1;
 if (Status && countA == 0 && countB == 1)
 {
  runningA = false;
  runningB = true;
  runningC = false;
  runningD = false;
  Question1(100);
 }
 else if (Status && countA == 1 && countB == 1 && runningA)
 {
  Start Variables();
  menu2();
 }
 else if (Status && countA == 0 && countB == 2 && runningB)
 {
  Start Variables();
  menu2();
 }
 else if (Status && countA == 0 && countC == 1 && countB == 3 && runningC)
 {
  Start Variables();
  menu2();
 }
 else if (Status && countA == 2 && countB == 1 && countC == 0)
 {
  Question2(" left valve.");
  runningD = false;
  LeftValve = true;
 }
 else if (Status && countA == 3 && countB == 1 && countC == 0 && RightValve)
 {
  menu3();
 }
 else if (Status && countA == 2 && countB == 2 && countC == 0 && LeftValve)
 {
  menu3();
 }
 else if (Status && countA == 2 && countB == 1 && countC == 1 && RightLeftValve)
 {
  menu3();
 }
 else if (Status && countA == 4 && countB == 1 && countC == 0 && RightValve)
 {
  Question4();
 }
 else if (Status && countA == 4 && countB == 2 && countC == 0 && RightValve)
 {
  countA = 4;
  countB = 0;
  countC = 0;
  menu4(" Right valve ");
 }
```

```
  else if (Status && countA == 5 && countB == 1 && countC == 0 && RightValve && SwitchR)
  {
   countA = 4;
   countB = 0;
   countC = 0;
   SwitchR = false;
   menu4(" Right valve ");
  }
  else if (Status && countA == 3 && countB == 2 && countC == 0 && LeftValve)
  {
   Question4();
  }
  else if (Status && countA == 3 && countB == 3 && countC == 0 && LeftValve)
  {
   countA = 3;
   countB = 1;
   countC = 0;
   menu4(" Left valve ");
  }
  else if (Status && countA == 4 && countB == 2 && countC == 0 && LeftValve && SwitchL)
  {
   countA = 3;
   countB = 1;
   countC = 0;
   SwitchL = false;
   menu4(" Left valve ");
  }
  else if (Status && countA == 3 && countB == 1 && countC == 1 && RightLeftValve)
  {
   Question4();
  }
  else if (Status && countA == 3 && countB == 2 && countC == 1 && RightLeftValve)
  {
   countA = 3;
   countB = 0;
   countC = 1;
   menu4(" Right and left valves ");
  }
  else if (Status && countA == 4 && countB == 1 && countC == 1 && RightLeftValve && SwitchRL)
  {
   countA = 3;
   countB = 0;
   countC = 1;
   SwitchRL = false;
   menu4(" Right and left valves ");
  }
  else
  {
   countB = countB - 1;
  }
  break;
  //////////////INSTRUCCIONES PARA EL ESCENARIO C////////////
 case 'c':
  countC = countC + 1;
  if (Status && countA == 0 && countC == 1 && countB == 0)
  {
    runningA = false;
    runningB = false;
    runningC = true;
    runningD = false;
    countB = 2;
    Question1(120);
  }
  else if (Status && countA == 2 && countB == 0 && countC == 1)
  {
    Question2(" right and Left valves.");
    runningD = false;
    RightLeftValve = true;
```

```
      }
      else
      {
       countC = countC - 1;
      }
      break;
      //////////////INSTRUCCIONES PARA EL ESCENARIO D////////////
     case 'd':
      //Serial.println(" Entro en el case d ");
      countD = countD + 1;
      if (Status && countA == 0 && countD == 1 && countC == 0 && countB == 0)
      {
       Status = false;
       menú();
      }
      else if (Status && countD == 1 && runningD)
      {
       Status = false;
       menu();
      }
      else
      {
       countD = countD - 1;
      }
      break;
      //////////////INSTRUCCIONES PARA EL ESCENARIO E////////////
     case 'e':
      if (!runningE)
      {
       runningE = true;
       menu();
      }
      else if (runningE)
      {
       runningE = false;
       StartVariables();
       StartValves();
       Status = false;
       Serial.println("Turning off..");
       delay(3000);
       digitalWrite(VoltValveRelief, LOW);
       Serial.println("Resetting valves ...");
       delay(3000);
       Serial.println("            ");
      }

      break;
      ////////////////// CIERRE DE ESCENARIOS //////////////////
     default:
      break;
    }
   }
  }
}
void StartVariables()
{
 countA = 0;
 countB = 0;
 countC = 0;
 countD = 0;
 p = 0;
 runningA = false;
 runningB = false;
 runningC = false;
 runningD = false;
 RightValve = false;
 LeftValve = false;
 RightLeftValve = false;
```

```
}
void StartVariables2()
{
 countA = 2;
 countB = 0;
 countC = 0;
 RightValve = false;
 LeftValve = false;
 RightLeftValve = false;
}
void StartValves()
{
 digitalWrite(VoltValveRight, LOW);
 digitalWrite(VoltValveLeft, LOW);
 digitalWrite(VoltValveRelief, HIGH);
}
void menu()
{
 Serial.println(" Starting...");
 delay(3000);
 StartVariables();
 StartValves();
 Serial.print("Supply pressure between 120 PSI and 140 PSI.");
 Serial.println("                                    1. Press (A) to confirm operation.");
}
void menu2()
{
 countA = 0;
 Serial.print("Select the operation pressure.              ");
 Serial.print("              1. Press (A) for 90 PSI.      ");
 Serial.print("                                  2. Press (B) for 100 PSI.       ");
 Serial.print("                                       3. Press (C) for 120 PSI.       ");
 Serial.println("                                          4. Press (D) for restore operation.");
}
void Question1(int pressure)
{
 Serial.print("Are you sure you want to use the pressure of ");
 Serial.print(pressure);
 Serial.print(" PSI ?");
 Serial.print("                                                1. Press (A) to confirm.          ");
 Serial.println("                                  2. Press (B) to cancel.          ");
}
void LoadedPressure(int LPressure)
{
 Serial.print("Loading ");
 Serial.print(LPressure);
 Serial.println(" PSI pressure...");
 delay(3000);
 p = LPressure;
 Serial.print(LPressure);
 Serial.println(" PSI loaded pressure");
 delay(3000);
 menu3();
}
void menu3()
{
 countA = 2;
 countB = 0;
 countC = 0;
 StartValves();
 Serial.print("Select valve to operate: ");
 Serial.print("                                    1. Press (A) to open right valve.");
 Serial.print("                                      2. Press (B) to open left valve.");
 Serial.print("                                      3. Press (C) to open right and left valves.");
 Serial.println("                                         4. Press (D) to restore operation.");
 runningD = true;
}
void Question2(String OpenValves)
```

```
{
 Serial.print("Are you sure you want to open");
 Serial.print(OpenValves       );
 Serial.print("                                                1. Press (A) to confirm.     ");
 Serial.println("                          2. Press (B) to Cancel.      ");
}
void Confirmation2(String ConfirmationValves2)
{
 Serial.print(ConfirmationValves2);
 Serial.println("opened ");
 delay(3000);
}
void menu4(String OpenValves2)
{
 Serial.print(OpenValves2);
 Serial.print("activated. ");
 Serial.print("                                  1. Press (A) to change the operation valve.     ");
 Serial.println("                  2. Press (B) to finish operation.              ");
}
void Question3()
{
 Serial.print("Are you sure you want to change the operating valve?");
 Serial.print("                                    1. Press (A) to confirm.     ");
 Serial.println("                                2. Press (B) to Cancel.      ");
}
void Question4()
{
 Serial.print("Are you sure you want to restore operation?");
 Serial.print("                                    1. Press (A) to confirm.     ");
 Serial.println("                                2. Press (B) to Cancel.      ");
}
```

Apéndice D

Código de programación de la tarjeta LaunchPad MSP - EXPA30G2ET para el equipo de apoyo en tierra.

```
float lectura;
float volt = 0;
float voltvalve;
int VoutConver;
int Vout = 19;              //----- Número de los pines de distribución electrónica.
int VoltValveRight = 11;
int VoltValveLeft = 12;
int VoltValveRelief = 13;
char letra;
bool runningA = false;
bool runningB = false;
bool runningC = false;
bool runningD = false;
bool runningE = false;
bool Status = false;
bool RightValve = false;
bool LeftValve = false;
bool RightLeftValve = false;
bool SwitchR = false;
bool SwitchL = false;
bool SwitchRL = false;
int p = 0;
int Cycle = 0;
int countA = 0;
int countB = 0;
int countC = 0;
int countD = 0;

////////ACTIVACIÓN Y DEFINICIÓN DEL LOS PINES///////
void setup()
{
 Serial.begin(9600);
 pinMode(A4, INPUT);              // ----Activación del pin A4 para la entrada de voltaje del conversor del sensor.
 pinMode(VoltValveRight, OUTPUT);
 pinMode(VoltValveLeft, OUTPUT);
 pinMode(VoltValveRelief, OUTPUT);
}
void loop()
{
 lectura = analogRead(A4);
 volt = lectura / 1023 *3.3;                    // Voltaje que entra a la tarjeta por el pin análogo de 0 - 3.3 v
 voltvalve = ( p * 3.3 * 3.3 ) / ( volt * 145 );    // Voltaje de salida al conversor de la válvula proporcional de 0 -  3.3 v
 VoutConver = (voltvalve * 255) / 3.3;
 analogWrite(Vout, VoutConver);
 if (Serial.available() > 0)
 {
  letra = Serial.read();
  lectura = analogRead(A4);
  volt = lectura / 1023 * 3.3;
  if ((!runningE &&& letra == 'e') || runningE)
```

```
{
 switch ( letra )
 {
  //////////////INSTRUCCIONES PARA EL EL ESCENARIO A////////////
  case 'a':
   countA = countA + 1;
   if (!Status && countA == 1)
   {
    Status = true;
    digitalWrite(VoltValveRelief, LOW);
    Serial.println(" Checking pressure...");
    delay(3000);
    if ( volt > 2.73 &&  volt < 6 )     // -------  Intervalo de voltaje para aceptar la presión de operación. para entrada de 0 - 3.3 v
    {
     Serial.println("Accepted pressure");
     delay(3000);
     digitalWrite(VoltValveRelief, HIGH);
     menu2();
    }
    else
    {
     Serial.println("Pressure not accepted");
     delay(3000);
     Serial.println("Restoring operation...");
     delay(1000);
     menu();
     Status = false;
    }
   }
   else if (countA == 1 && Status && !runningA && countB == 0 && countC == 0)
   {
    runningA = true;
    runningB = false;
    runningC = false;
    runningD = false;
    Question1(90);
   }
   else if (Status && countA == 2 && runningA)
   {
    LoadedPressure(90);
   }
   else if (Status && countA == 1 && countB == 1 && runningB)
   {
    LoadedPressure(100);
   }
   else if (Status && countA == 1 && countC == 1 && runningC)
   {
    LoadedPressure(120);
   }
   else if (Status && countA == 3 && countB == 0 && countC == 0)
   {
    Question2(" right valve.");

    RightValve = true;
    runningD = false;
   }
   else if (Status && countA == 4 && countB == 0 && countC == 0 && RightValve)
   {
    Serial.println("Opening right valve...");
    delay(3000);
    digitalWrite(VoltValveRight, HIGH);
    Confirmation2(" Right valve ");
    menu4(" Right valve ");
   }
   else if (Status && countA == 3 && countB == 1 && countC == 0 && LeftValve)
   {
    Serial.println("Opening left valve...");
    delay(3000);
```

```
 digitalWrite(VoltValveLeft, HIGH);
 Confirmation2(" Left valve ");
 menu4(" Left valve ");
}
else if (Status && countA == 3 && countB == 0 && countC == 1 && RightLeftValve)
{
 Serial.println("Opening right and left valves...");
 delay(3000);
 digitalWrite(VoltValveLeft, HIGH);
 digitalWrite(VoltValveRight, HIGH);
 Confirmation2(" Right and left valves ");
 menu4(" Right and left valve ");
}
else if (Status && countA == 5 && countB == 0 && countC == 0 && RightValve)
{
 Question3();
 SwitchR = true;
}
else if (Status && countA == 6 && countB == 0 && countC == 0 && RightValve)
{
 Serial.println("Closing right valve...");
 delay(3000);
 StartValves();
 Serial.println("Right valve closed");
 delay(3000);
 RightValve = false;
 StartVariables2();
 menu3();
}
else if (Status && countA == 5 && countB == 1 && countC == 0 && RightValve && !SwitchR)
{
 Serial.println("Restoring operation...");
 delay(3000);
 menu();
}
else if (Status && countA == 4 && countB == 1 && countC == 0 && LeftValve)
{
 Question3();
 SwitchL = true;
}
else if (Status && countA == 5 && countB == 1 && countC == 0 && LeftValve)
{
 Serial.println("Closing left valve...");
 delay(3000);
 StartValves();
 Serial.println("Left valve closed");
 delay(3000);
 LeftValve = false;
 StartVariables2();
 menu3();
}
else if (Status && countA == 4 && countB == 2 && countC == 0 && LeftValve && !SwitchL)
{
 Serial.println("Restoring operation...");
 delay(3000);
 menu();
}
else if (Status && countA == 4 && countB == 0 && countC == 1 && RightLeftValve)
{
 Question3();
 SwitchRL = true;
}
else if (Status && countA == 5 && countB == 0 && countC == 1 && RightLeftValve)
{
 Serial.println("Closing right and left valves...");
 delay(3000);
 StartValves();
 Serial.println("Right and left valves closed");
```

```
  delay(3000);
  RightLeftValve = false;
  StartVariables2();
  menu3();
 }
 else if (Status && countA == 4 && countB == 1 && countC == 1 && RightLeftValve && !SwitchRL)
 {
  Serial.println("Restoring operation...");
  delay(3000);
  menu();
 }
 else
 {
  countA = countA - 1;
 }
 break;
//////////////INSTRUCCIONES PARA EL EL ESCENARIO B//////////////
case 'b':
 //Serial.println(" Entro en el case b ");
 countB = countB + 1;
 if (Status && countA == 0 && countB == 1)
 {
  runningA = false;
  runningB = true;
  runningC = false;
  runningD = false;
  Question1(100);
 }
 else if (Status && countA == 1 && countB == 1 && runningA)
 {
  StartVariables();
  menu2();
 }
 else if (Status && countA == 0 && countB == 2 && runningB)
 {
  StartVariables();
  menu2();
 }
 else if (Status && countA == 0 && countC == 1 && countB == 3 && runningC)
 {
  StartVariables();
  menu2();
 }
 else if (Status && countA == 2 && countB == 1 && countC == 0)
 {
  Question2(" left valve.");
  runningD = false;
  LeftValve = true;
 }
 else if (Status && countA == 3 && countB == 1 && countC == 0 && RightValve)
 {
  menu3();
 }
 else if (Status && countA == 2 && countB == 2 && countC == 0 && LeftValve)
 {
  menu3();
 }
 else if (Status && countA == 2 && countB == 1 && countC == 1 && RightLeftValve)
 {
  menu3();
 }
 else if (Status && countA == 4 && countB == 1 && countC == 0 && RightValve)
 {
  Question4();
 }
 else if (Status && countA == 4 && countB == 2 && countC == 0 && RightValve)
 {
  countA = 4;
```

```
  countB = 0;
  countC = 0;
  menu4(" Right valve ");
 }
 else if (Status && countA == 5 && countB == 1 && countC == 0 && RightValve && SwitchR)
 {
  countA = 4;
  countB = 0;
  countC = 0;
  SwitchR = false;
  menu4(" Right valve ");
 }
 else if (Status && countA == 3 && countB == 2 && countC == 0 && LeftValve)
 {
  Question4();
 }
 else if (Status && countA == 3 && countB == 3 && countC == 0 && LeftValve)
 {
  countA = 3;
  countB = 1;
  countC = 0;
  menu4(" Left valve ");
 }
 else if (Status && countA == 4 && countB == 2 && countC == 0 && LeftValve && SwitchL)
 {
  countA = 3;
  countB = 1;
  countC = 0;
  SwitchL = false;
  menu4(" Left valve ");
 }
 else if (Status && countA == 3 && countB == 1 && countC == 1 && RightLeftValve)
 {
  Question4();
 }
 else if (Status && countA == 3 && countB == 2 && countC == 1 && RightLeftValve)
 {
  countA = 3;
  countB = 0;
  countC = 1;
  menu4(" Right and left valves ");
 }
 else if (Status && countA == 4 && countB == 1 && countC == 1 && RightLeftValve && SwitchRL)
 {
  countA = 3;
  countB = 0;
  countC = 1;
  SwitchRL = false;
  menu4(" Right and left valves ");
 }
 else
 {
  countB = countB - 1;
 }
 break;
//////////////INSTRUCCIONES PARA EL EL ESCENARIO C/////////////
case 'c':
 countC = countC + 1;
 if (Status && countA == 0 && countC == 1 && countB == 0)
 {
  runningA = false;
  runningB = false;
  runningC = true;
  runningD = false;
  countB = 2;
  Question1(120);
 }
 else if (Status && countA == 2 && countB == 0 && countC == 1)
```

```
      {
       Question2(" right and Left valves.");
       runningD = false;
       RightLeftValve = true;
      }
      else
      {
       countC = countC - 1;
      }
      break;
     //////////////INSTRUCCIONES PARA EL EL ESCENARIO D//////////////
     case 'd':
      //Serial.println(" Entro en el case d ");
      countD = countD + 1;
      if (Status && countA == 0 && countD == 1 && countC == 0 && countB == 0)
      {
       Status = false;
       menu();
      }
      else if (Status && countD == 1 && runningD)
      {
       Status = false;
       menu();
      }
      else
      {
       countD = countD - 1;
      }
      break;
     //////////////INSTRUCCIONES PARA EL EL ESCENARIO E//////////////
     case 'e':
      if (!runningE)
      {
       runningE = true;
       menu();
      }
      else if (runningE)
      {
       runningE = false;
       StartVariables();
       StartValves();
       Status = false;
       Serial.println("Turning off..");
       delay(3000);
       digitalWrite(VoltValveRelief, LOW);
       Serial.println("Resetting valves ...");
       delay(3000);
       Serial.println("            ");
      }
      break;
     ////////////////////// CIERRE DE ESCENARIOS ///////////////////
     default:
      break;
    }
   }
  }
}
void StartVariables()
{
 countA = 0;
 countB = 0;
 countC = 0;
 countD = 0;
 p = 0;
 runningA = false;
 runningB = false;
 runningC = false;
 runningD = false;
```

```
 RightValve = false;
 LeftValve = false;
 RightLeftValve = false;
}
void StartVariables2()
{
 countA = 2;
 countB = 0;
 countC = 0;
 RightValve = false;
 LeftValve = false;
 RightLeftValve = false;
}
void StartValves()
{
 digitalWrite(VoltValveRight, LOW);
 digitalWrite(VoltValveLeft, LOW);
 digitalWrite(VoltValveRelief, HIGH);
}
void menu()
{
 Serial.println(" Starting...");
 delay(3000);
 StartVariables();
 StartValves();
 Serial.print("Supply pressure between 120 PSI and 140 PSI.");
 Serial.println("                                    1. Press (A) to confirm operation.");
}
void menu2()
{
 countA = 0;
 Serial.print("Select the operation pressure.              ");
 Serial.print("              1. Press (A) for 90 PSI.      ");
 Serial.print("                                  2. Press (B) for 100 PSI.       ");
 Serial.print("                                       3. Press (C) for 120 PSI.       ");
 Serial.println("                                          4. Press (D) for restore operation.");
}
void Question1(int pressure)
{
 Serial.print("Are you sure you want to use the pressure of ");
 Serial.print(pressure);
 Serial.print(" PSI ?");
 Serial.print("                                              1. Press (A) to confirm.            ");
 Serial.println("                                  2. Press (B) to cancel.          ");
}
void LoadedPressure(int LPressure)
{
 Serial.print("Loading ");
 Serial.print(LPressure);
 Serial.println(" PSI pressure...");
 delay(3000);
 p = LPressure;
 Serial.print(LPressure);
 Serial.println(" PSI loaded pressure");
 delay(3000);
 menu3();
}
void menu3()
{
 countA = 2;
 countB = 0;
 countC = 0;
 StartValves();
 Serial.print("Select valve to operate: ");
 Serial.print("                                    1. Press (A) to open right valve.");
 Serial.print("                                      2. Press (B) to open left valve.");
 Serial.print("                                      3. Press (C) to open right and left valves.");
 Serial.println("                                         4. Press (D) to restore operation.");
```

```
 runningD = true;
}
void Question2(String OpenValves)
{
 Serial.print("Are you sure you want to open");
 Serial.print(OpenValves       );
 Serial.print("                                                1. Press (A) to confirm.     ");
 Serial.println("                          2. Press (B) to Cancel.      ");
}
void Confirmation2(String ConfirmationValves2)
{
 Serial.print(ConfirmationValves2);
 Serial.println("opened ");
 delay(3000);
}
void menu4(String OpenValves2)
{
 Serial.print(OpenValves2);
 Serial.print("activated. ");
 Serial.print("                                1. Press (A) to change the operation valve.     ");
 Serial.println("                  2. Press (B) to finish operation.              ");
}
void Question3()
{
 Serial.print("Are you sure you want to change the operating valve?");
 Serial.print("                                    1. Press (A) to confirm.     ");
 Serial.println("                                2. Press (B) to Cancel.      ");
}
void Question4()
{
 Serial.print("Are you sure you want to restore operation?");
 Serial.print("                                    1. Press (A) to confirm.     ");
 Serial.println("                                2. Press (B) to Cancel.      ");
}
```

Apéndice E

Código de programación del panel de control en el software processing para el equipo de apoyo en tierra.

```
import processing.serial.*;
  boolean EstadoBotonA = false;
  Serial MiSerial;
  String dato="";
  PImage img;
void setup() {
   size(1200, 650);
   String NombrePuerto ="COM4";      // -------- Selección del puerto de comunicación de la tarjeta.
   MiSerial = new Serial(this, NombrePuerto, 9600);
    img = loadImage("manometro.png");    // -------- Selección de imagen de manómetro para ilustración.
}
void draw () {
   background(256);
   image(img, 85, 350, 250, 250);
   image(img, 865, 350, 250, 250);
   fill(0, 0, 204);
   rect(400, 150, 400, 180);
   fill(255, 152, 0);
   ellipse(450, 400, 40, 40);
   ellipse(550, 400, 40, 40);
   ellipse(650, 400, 40, 40);
   ellipse(750, 400, 40, 40);
   ellipse(750, 400, 40, 40);
   fill(0, 255, 0);
   ellipse(600, 520, 40, 40);
   textAlign(CENTER);
   textSize(50);
   fill(242, 243, 244);
   text("PNEUMATIC PRESSURE TEST EQUIPMENT", 600, 100);
   textAlign(CENTER);
   textSize(30);
   text("A       B       C       D", 600, 465 );
   textAlign(CENTER);
   textSize(20);
   text("ON/OFF", 600, 580 );
   textAlign(CENTER);
   textSize(20);
   fill(242, 243, 244);
   text("CONDITIONS", 200, 170);
   textSize(10);
   text("1. Use only following the instructions in the maintenance manual.", 200, 200);
   text("2. 115 volt AC electrical power supply.", 200, 220);
   text("3. The inlet pressure range for testing is 120 to 140 PSI.", 200, 240);
   text("4. Gauge 1 reads the initial pressure.", 200, 260);
   text("5. Gauge 2 reads the operating pressure loaded in the system.", 200, 280);
   textSize(20);
   text("WARNING!", 1000, 170);
   textSize(10);
   text("1. Do not exceed initial pressure above 145 PSI.", 1000, 200);
   text("2. Do not disconnect the pressure lines in the test.", 1000, 220);
   text("3. Turn off the equipment if you want to end the test.", 1000, 240);
   text("4. If gauge two does not indicate the operating pressure, ", 1000, 260);
   text("the equipment must be calibrated.", 1000, 280);
```

```
    if (dato != null){
     textAlign(LEFT);
     textSize(16);
     fill(242, 243, 244);
     text(dato, 410, 160, 380, 150);
    }
}
void mouseClicked(){
 float distancia = dist(450, 400, mouseX, mouseY);
 if(distancia < 20) {
 MiSerial.write('a');
 }
 float distancia1 = dist(550, 400, mouseX, mouseY);
 if(distancia1 < 20) {
 MiSerial.write('b');
 }
 float distancia2 = dist(650, 400, mouseX, mouseY);
 if(distancia2 < 20) {
 MiSerial.write('c');
 }
 float distancia3 = dist(750, 400, mouseX, mouseY);
 if(distancia3 < 20) {
 MiSerial.write('d');
 }
 float distancia4 = dist(600, 520, mouseX, mouseY);
 if(distancia4 < 20) {
 MiSerial.write('e');
 }
}
void serialEvent (Serial MiSerial){
  dato = MiSerial.readStringUntil('\n');
}
```