DISEÑO Y CONSTRUCCIÓN DE UNA INTERFAZ HARDWARE PARA UN SIMULADOR DE VUELO

Juan Fernando Cely Ortega

Universidad del Cauca
Facultad de Ciencias Naturales Exactas y de la Educación
Ingeniería Física
Popayán
2019

DISEÑO Y CONSTRUCCIÓN DE UNA INTERFAZ HARDWARE PARA UN SIMULADOR DE VUELO

Juan Fernando Cely Ortega

Trabajo de Grado para optar por el título de Ingeniero Físico

Ing. Mario Andrés Córdoba Gonzales

Universidad del Cauca
Facultad de Ciencias Naturales Exactas y de la Educación
Ingeniería Física
Popayán
2019

Nota de Aceptación

_____

_____

_____

_____

Director del trabajo de grado.

_____

Jurado

_____

Jurado

Popayán, Cauca, 7 de febrero de 2019

CONTENIDO

CAPITULO 4: ADQUSICIÓN Y PROCESAMIENTO DE LA INFORMACION

4.1 Configuración del entorno de programación de la tarjeta de desarrollo beagle bone black.
4.1.1 Instalación del sistema operativo: DEBIAN.
4.2 Configuración del entorno de programación de la tarjeta de desarrollo STM32f4 núcleo.
4.2.1 Configuración del IDE de desarrollo.
4.3 software de interfaz LINK2FS.
4.3.1 Configuración de parámetros de salida de link2fs para una aeronave Cessna 172.
4.3.2 Configuración de parámetros de entrada de link2fs para una aeronave Cessna 172.
4.4 Software para interfaz de indicadores: Air Manager.
4.5 Scripts de programación para cada componente electrónico.

CAPITULO 5: RESULTADOS.

5.1 Resultados en la construcción de cada módulo.
5.1.1 Modulo de potencia.
5.1.2 Modulo de botones.
5.1.3 Modulo de comunicaciones.
5.2 Conexión final.
5.3 Pruebas de los módulos construidos.
5.4 Costos del demo de la aeronave Cessna 172.

CONCLUSIONES.

TRABAJOS FUTUROS.

BIBLIOGRAFIA.

ANEXOS.

A- Anexo de la configuración de IDE de desarrollo.
B- Anexo de los scripts de programación del código de cada componente.
C- Anexo de los esquemáticos diseñados para la elaboración del circuito.

LISTA DE TABLAS

LISTA DE FIGURAS

INTRODUCCIÓN

La formación de un piloto de avión requiere el desarrollo de diferentes habilidades como la navegación, pilotaje y manejo de instrumentación. Son muchos los detalles que un piloto debe considerar durante el vuelo, por lo que sus primeras experiencias resultan peligrosas sin el entrenamiento adecuado. Para facilitar esta tarea a inicios del siglo XX se empezaron a desarrollar cabinas articuladas que simularan la experiencia que se tiene al pilotar un avión, esto debido a la necesidad de formar pilotos en la Primera Guerra Mundial. Un resultado de aquella época es el entrenador conocido como *Link Flight Trainer* [1], originalmente desarrollado en 1929 como prototipo, consiste en una pequeña cabina en forma de avión que contiene todos los instrumentos y controles básicos en un avión de ese entonces. Las acciones del piloto eran respondidas por un sistema electro mecánico que a través de pistones y motores simulaban las reacciones del avión.

Un simulador de vuelo hoy se define como el conjunto de sistemas que intentan simular de la manera más precisa y realista la experiencia de controlar, manejar y pilotear una aeronave. Algunos simuladores van desde simples videojuegos hasta réplicas exactas de cabinas (a escala o tamaño real) de accionamiento eléctrico, neumático, mecánico, etc. Controlados bajo sistemas computarizados. Entidades como la Administración Federal de Aviación de Estados Unidos (FAA) propone estándares y certifica simuladores de vuelo para organizaciones que entrenan pilotos en su regulación FAR 121, esto muestra la importancia de estos equipos en la formación de un piloto. [2]

Actualmente se requiere un componente software y otro hardware en la construcción de un simulador de vuelo, respecto al primero, existen varias alternativas aprobadas por la FAA como: Microsoft Flight Simulator, X-Plane, Redbird Flight, FLYIT, entre otros. El hardware puede ser, desde un sistema ajustado a configuraciones tan sencillas como teclado y un ratón hasta la réplica de una cabina, lo anterior se explica mediante el software que ofrece soporte a extensiones que permitan realizar implementaciones de hardware a medida.

La aparición de las tarjetas de desarrollo y el crecimiento de la comunidad *maker* ha reducido los costos de estas soluciones de hardware, acompañada de una extensa documentación para aprovechar sus funcionalidades. Estas herramientas facilitan el diseño para una interfaz que cuente con protocolos de comunicación, conexiones de entradas y salidas para crear controles, mandos y visualizadores de un simulador, haciendo el proceso más fácil y con menores costos. [3]

Se abordará el primer capítulo las generalidades teóricas necesarias para comprender la temática desarrollada, un segundo capítulo donde se abordará del diseño y componentes para la construcción de un simulador básico, seguido de un capítulo donde se dará a conocer la adquisición y procesamiento de todos los datos para posteriormente finalizar abordando el capítulo donde se darán a conocer los resultados obtenidos.

OBJETIVOS

OBJETIVO GENERAL

- Diseñar y construir la interfaz hardware de los sistemas de instrumentación de un simulador de vuelo para una aeronave CESSNA 172.

OBJETIVOS ESPECÍFICOS

- Definir las herramientas de software y el protocolo de comunicaciones entre el simulador de vuelo y el hardware a construir.

- Construir los módulos de instrumentos, botones y actuadores requeridos por el simulador.

- Calibrar y poner a punto todos los módulos implementados.

PLANTEAMIENTO DEL PROBLEMA

En la actualidad, los sistemas de simulación tipo software – hardware permiten la interfaz de unos cuantos periféricos básicos (Joystick, Controles, Pedales, etc.) que no están estandarizados, lo cual no permite una simulación real de alguna aeronave en específico. Además, la simulación de palancas, botones y actuadores hay que realizarlas de manera poco intuitiva, ya que se deben seleccionar manualmente sobre el software o asignarles una tecla en específico en un teclado; existen otros sistemas propietarios a nivel comercial que proporcionan la aviónica (sistemas de instrumentación) a costos muy elevados, quedando fuera del alcance de las escuelas de entrenamiento en países como Colombia. [4]

De acuerdo a esta problemática (costos e interfaz), se formula la pregunta de investigación para este trabajo ¿es posible realizar un simulador a bajo costo de categoría A de una aeronave Cessna 172, que tenga las funcionalidades básicas como radios, transponder, piloto automático, botones de iluminación, sistemas de potencia e indicadores de instrumentos, que pueda procesar toda la información recibida y enviada sin perder ninguna trama de datos tal como los que poseen las escuelas de aviación colombianas?

JUSTIFICACION

Las escuelas de aviación regionales no cuentan con hardware apropiado para sus simuladores de vuelo, emplean equipos que se consiguen en el mercado que no ofrecen ningún tipo de opción para ajustarlos a las necesidades que existen en el entrenamiento. Se plantea en este trabajo de investigación el desarrollo de una solución modular y ajustable que permitirá a las escuelas de aviación ofrecer un entorno de simulación más real que tenga la opción de adicionar los módulos necesarios para ser escalable, es decir, que sea lo más parecido a la cabina de la aeronave a simular, entre más parecido sea, el piloto en entrenamiento podrá hacer muchas de las tareas de control y comunicación de manera natural y fluida. Lo que se quiere con esta investigación, es que lo dispositivos costosos y poco robustos que se encuentran fácilmente en el mercadeo tiendan a ser desplazados por equipos que puedan suplir las necesidades al 100% de las escuelas de aviación.

Adicionalmente, parte de los componentes eléctricos y mecánicos se pueden conseguir construir dentro del país, permitiendo la reducción de costos y favoreciendo a los pilotos en formación que cada pudiera tener su simulador propio para que puedan acumular muchas más horas de vuelo en simulador por la misma o menor relación de costo que tienen actualmente.

CAPITULO 1: FUNDAMENTOS TEÓRICOS.

El funcionamiento de algunos de los instrumentos está basado en la presión del aire, giroscopios y comunicaciones. La idea es recordar estos conceptos con el fin de entender en el siguiente capítulo como es el funcionamiento que hay internamente en los instrumentos con los que el piloto tiene interacción todo el tiempo en el vuelo.

1.1 Instrumentos de presión.

La atmósfera de la Tierra es continua desde el suelo hasta la frontera con el espacio, pero se puede describir en términos de capas a diferentes altitudes, cada una tiene propiedades distintas. Una de las propiedades que distingue las capas, es la forma en que la temperatura varía con la altitud, como se muestra en la Figura 1.1. En La troposfera, que es la capa más baja junto al suelo, la temperatura disminuye linealmente con la altura hasta la tropopausa, que marca el límite entre la troposfera y la siguiente capa. En el fondo de la estratosfera, la temperatura permanece constante con la altitud, pero por encima de 20 km la temperatura vuelve a bajar y alcanza un mínimo. Dentro de la mesosfera a temperatura comienza a aumentar de nuevo. La capa sobre la mesosfera es conocida como la termosfera que comienza a los 85 km y está al límite con el espacio. [4]

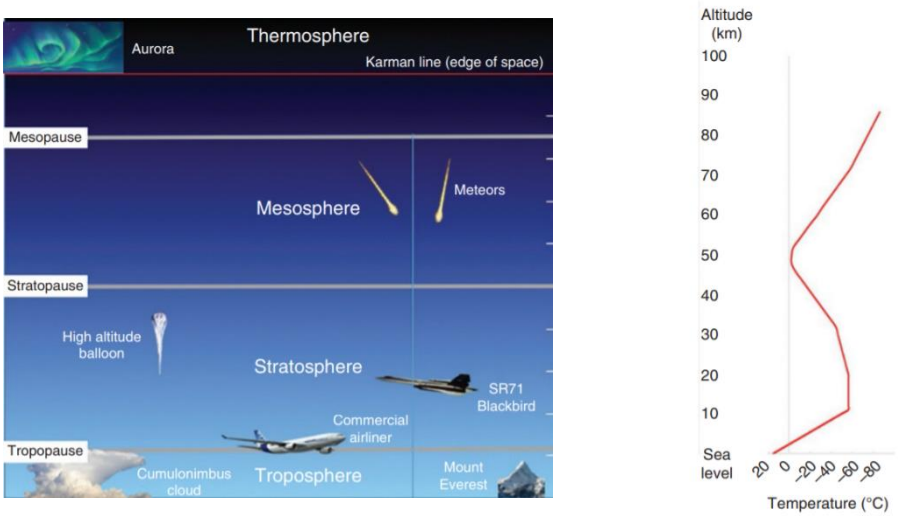

Figura 1.1 División de la atmosfera terrestre y variación de la temperatura con respecto a altitud.

Generalmente se acepta estar en la línea de Karman a una altitud de 100 km. Esto lleva el nombre del físico Theodore Karman, quien fue el primero en demostrar que por encima de esta línea una aeronave tendría que viajar más rápido que la velocidad orbital para generar suficiente sustentación aerodinámica para mantener la altitud. Así, el vuelo es imposible, incluso en principio, por encima de esta altura, aunque en la práctica se limita a menos de unos 30 km. Las aeronaves se limitan a las dos capas inferiores de la atmósfera, es decir, las la troposfera y la estratosfera. Los aviones de transporte comercial en general, vuela a la tropopausa, los aviones supersónicos en la estratosfera y los aviones de hélice en la

troposfera. Prácticamente todo el clima se limita a la troposfera, que Contiene el 75% de la masa de la atmósfera y el 99% del vapor de agua. [4]

1.2 Definición de nivel del mar y elevación.

Para definir la altura a la que vuela un avión se requiere una referencia en la Tierra. La superficie y una elección natural es el nivel del mar, ya que la gravedad atrae el agua hacia el centro de la tierra, por lo que todo debería establecerse en un nivel común. Por definición, la altura de la topografía del suelo u objeto hecho por el hombre sobre el nivel del mar es conocida como su elevación. Así, se nos dice que la elevación de la cima del monte Everest es de 29.002 pies (8.848 m) o la elevación del aeródromo local es de 469 pies (143 m). Sin embargo, definir alturas precisas en la navegación global, requiere un mayor cuidado considerando de lo que realmente significa el nivel del mar y cómo se determina. [4]

1.3 Definición de altura, altitud y nivel de vuelo.

Antes de describir los instrumentos de presión es importante recordar las definiciones de altura, altitud y elevación, donde todas se refieren a distancias verticales, pero tienen significados definidos en la aviación como lo muestra la siguiente gráfica:

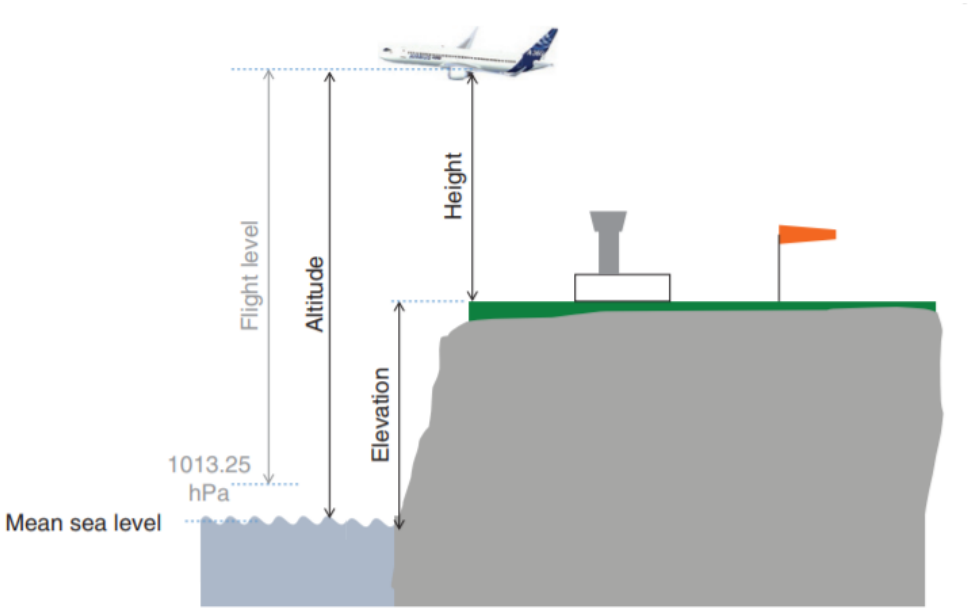

Figura 1.2 Diferencia entre altura, altitud y nivel de vuelo.

La elevación se refiere a la distancia vertical sobre el nivel del mar de una característica del terreno fijo, que puede ser un terreno natural o una estructura artificial como un mástil. La altitud se define como la distancia vertical de un avión que vuela sobre el nivel del mar, mientras que la altura es la distancia vertical sobre un dato definido por el usuario, como una superficie de pista.

Loss instrumentos de presión, es decir, el altímetro, el indicador de velocidad vertical, el indicador de velocidad del aire y el medidor de Mach requieren sondas que midan la presión estática y dinámica. [4]

La presión es la magnitud escalar que relaciona la fuerza con la superficie sobre la cual actúa; es decir, equivale a la fuerza que actúa sobre la superficie. Cuando sobre una superficie plana de área A se aplica una fuerza normal F de manera uniforme, la presión P viene dada de la siguiente forma:

$$p = \frac{F}{A}$$

La presión estática es la que tiene un fluido, independientemente de la velocidad del mismo, y que se puede medir mediante la utilización de tubos piezométricos.

Se puede decir que cuando los fluidos se mueven en un conducto, la inercia del movimiento provoca un incremento adicional de la presión estática al chocar sobre un área perpendicular al movimiento. La presión dinámica depende de la velocidad y la densidad del fluido, definida en la siguiente ecuación:

$$q = \frac{1}{2}\rho v^2$$

La presión total que ejerce un fluido -bien sea gaseoso o líquido- se define como la suma de la presión estática y la presión dinámica.

La sonda de presión estática puede ser, en principio, solo un agujero al aire exterior, pero requiere una cuidadosa colocación en la estructura del avión, internamente su funcionamiento seria como lo muestra la siguiente gráfica, debido a la presión que entre en el tubo, esta moverá el sistema mecánico para ser visualizado en este caso en el indicador de velocidad.

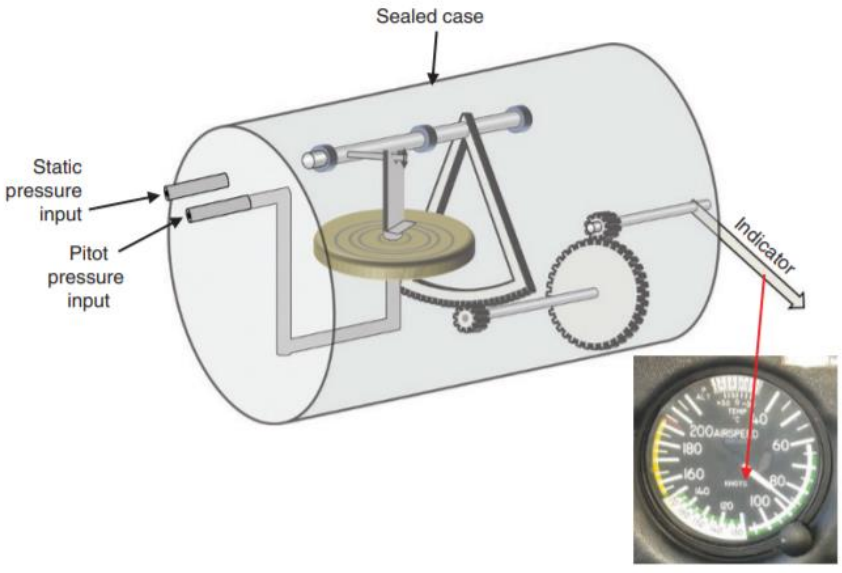

Figura 1.3: mecanismo interno de los instrumentos que funcionan con el cambio de presión.

1.4 Propiedades básicas de los giroscopios mecánicos

Hay dos propiedades de los giroscopios que se utilizan en la aviación, es decir, rigidez y precesión. La rigidez es la tendencia del eje de giro a permanecer. Se fija en el espacio inercial y, en la práctica, se puede considerar que el eje se mantiene en relación con una estrella distante como se muestra en la figura 1.4 La rigidez aumenta con El momento angular del giroscopio, que es dado por:

$$L = I * \omega$$

Aquí es el momento de inercia y ω es la velocidad angular. Las variables (L, ω) son cantidades vectoriales, donde la dirección de los vectores es a lo largo del eje de rotación, mientras que I es un escalar. [4]

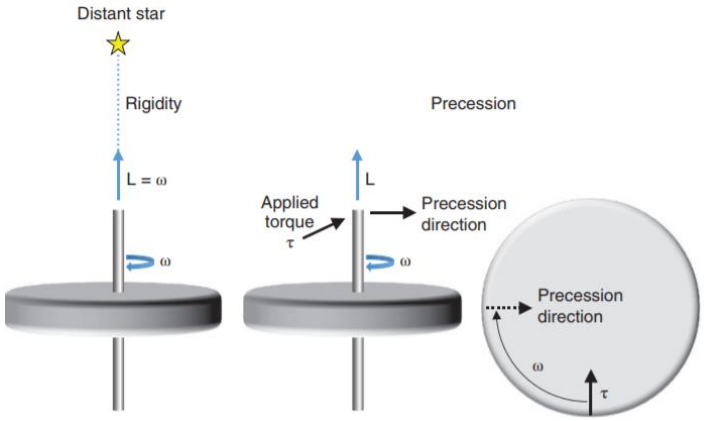

Figura 1.4: propiedades básicas de los giroscopios.

Se usa el giroscopio en los instrumentos que requieren que este mantenga una constante dirección de giro como el indicador de dirección.

Hay varias opciones de conjuntos cartesianos de ejes, incluido un sistema basado en la Tierra y un sistema de trayectoria de aire, pero el conjunto que se utiliza se denomina eje del cuerpo sistema y se ilustra en la figura 1.5 Todos los ejes pasan por el centro de
la gravedad de la aeronave y se fijan en relación con la estructura del avión, por lo que giran con la aeronave. El eje longitudinal es paralelo a la línea central del fuselaje, el lateral el eje es ortogonal a lo largo de las alas, y el eje vertical es ortogonal a ambos. Se supone que el avión es un cuerpo rígido y no se dobla durante las rotaciones. Las rotaciones sobre los ejes utilizan la misma terminología que se ha utilizado en barcos desde la antigüedad. Rodar es una rotación alrededor del eje longitudinal, la inclinación es una rotación alrededor del eje lateral, y la orientación es una rotación alrededor del eje vertical. Un conjunto dado de balanceo, cabeceo y desvío.
Los valores se conocen como la actitud de la aeronave. Los diversos instrumentos muestran. la inclinación de balanceo y la guiñada para que el piloto tenga una imagen de la actitud del avión sin ninguna referencia externa. [4]

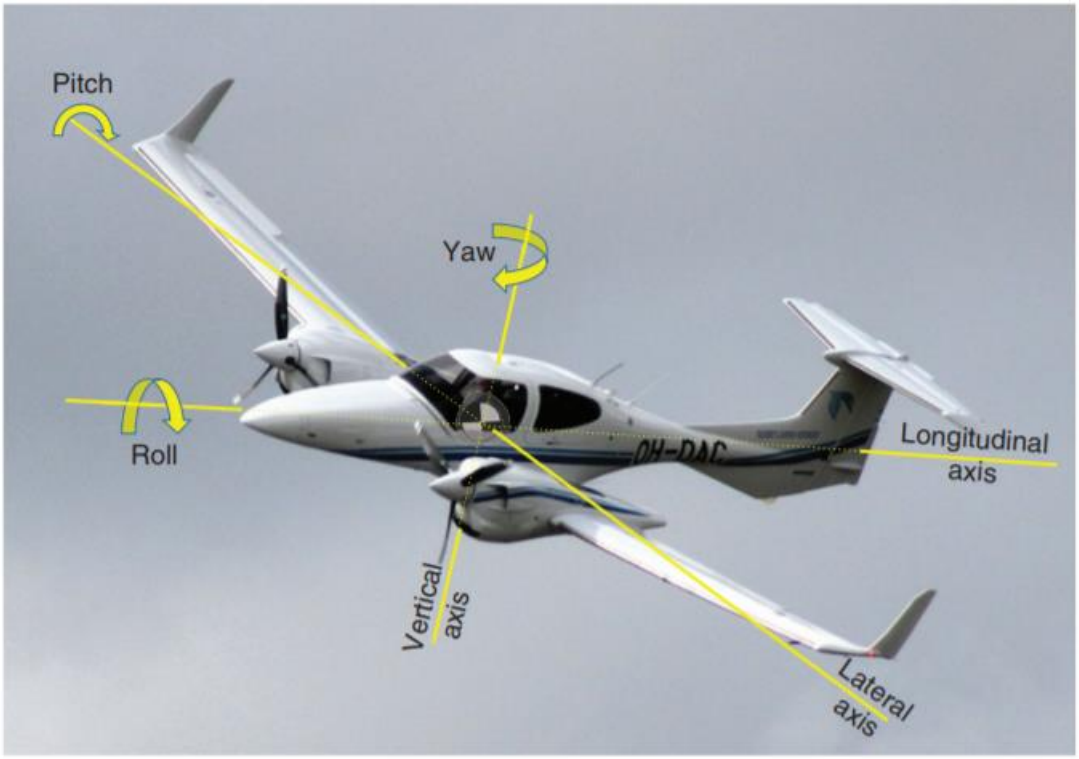

Figura 1.5: ejes de rotación de la aeronave.

El indicador de ruta, tiene un giroscopio con su eje en el plano horizontal en un conjunto de doble cardán (dos grados de libertad) para que Puede girar libremente alrededor de dos ejes, tanto perpendiculares al eje de giro. La caja puede ser bloqueado en el plano horizontal y el cardán externo girado de manera que la tarjeta muestre la dirección indicada de la brújula. El cardán puede entonces se desbloqueará y el eje horizontal se mantendrá de modo que cuando la aeronave gira, una unidad de engranaje mueve un puntero para indicar la nueva dirección en la tarjeta visible en la cabina del piloto. [4]

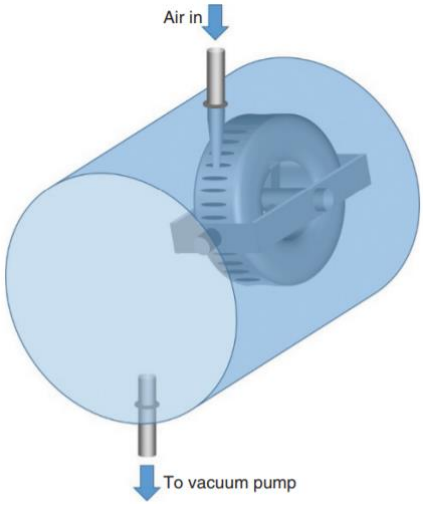

Figura 1.6: bomba de vacío para el funcionamiento del giroscopio.

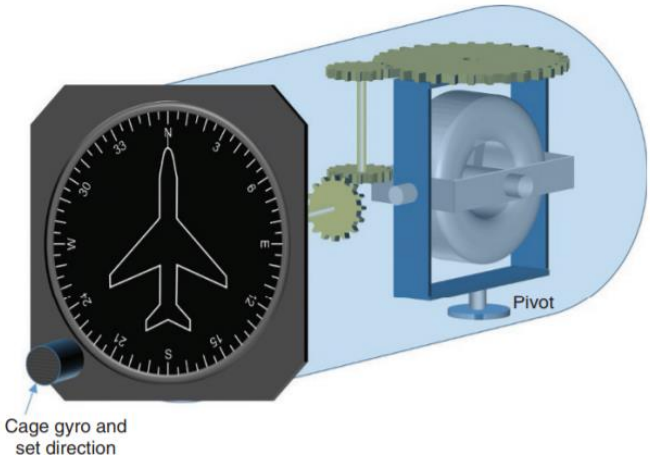

Figura 1.7: funcionamiento interno de los instrumentos que funcionan con el giroscopio.

1.5 Propagación de radio y comunicaciones.

Según lo verificado por Heinrich Hertz en 1886, las ondas de radio son ondas electromagnéticas. Viajan a la velocidad de la luz y son la propagación de un oscilante eléctrico y el campo magnético orientado perpendicular entre sí como se muestra en la figura 1.7. Este es un ejemplo de una onda polarizada linealmente, es decir, la dirección del campo eléctrico permanece fijo y la dirección de polarización se define a lo largo del campo eléctrico. Este tipo de onda es emitida por un Antena dipolo simple. Otro tipo de polarización de las ondas de radio utilizadas en la aviación se conoce como una onda polarizada circularmente. En este caso, la dirección del campo eléctrico gira una vez por ciclo de variación de amplitud, de modo que el observador podría detectar la dirección del campo eléctrico que giraría al mismo tiempo. Diferentes polarizaciones se utilizan en la aviación dependiendo de la Aplicación y en el caso de algunas aplicaciones de radar la polarización detectada, es diferente al emitido para resaltar características específicas. La relación entre la frecuencia y la longitud de onda electromagnético es:

$$f = \frac{c}{\lambda}$$

donde c es la velocidad de la luz y λ la longitud de onda. [4]

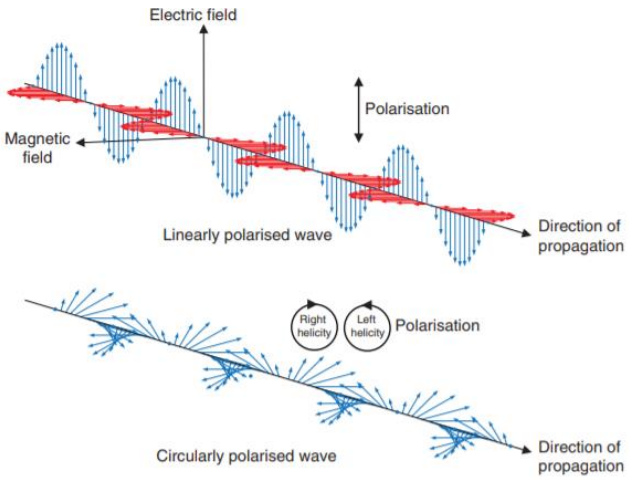

Figura 1.7 desplazamiento de las ondas electromagnéticas.

Las ondas de radio utilizadas en la aviación cubren un gran espectro de frecuencias de 30 kHz a 40 GHz. Este espectro se subdivide en bandas y la clasificación más utilizada es la definida por la Union de Telecomunicaciones Internacional (UIT), que es una rama especializada de la ONU. La figura 1.8 muestra los nombres de las bandas de frecuencia definidas por la UIT y también Indica su papel en la aviación civil. Otro esquema de clasificación definido por El Instituto de Ingenieros Eléctricos y Electrónicos (IEEE) también se utiliza, especialmente para las frecuencias más altas. El esquema IEEE es el mismo que el especificado por la UIT hasta la banda UHF, pero subdivide la más alta frecuencias en las bandas C, L, S, X y K como se indica en la figura 1.8. [4]

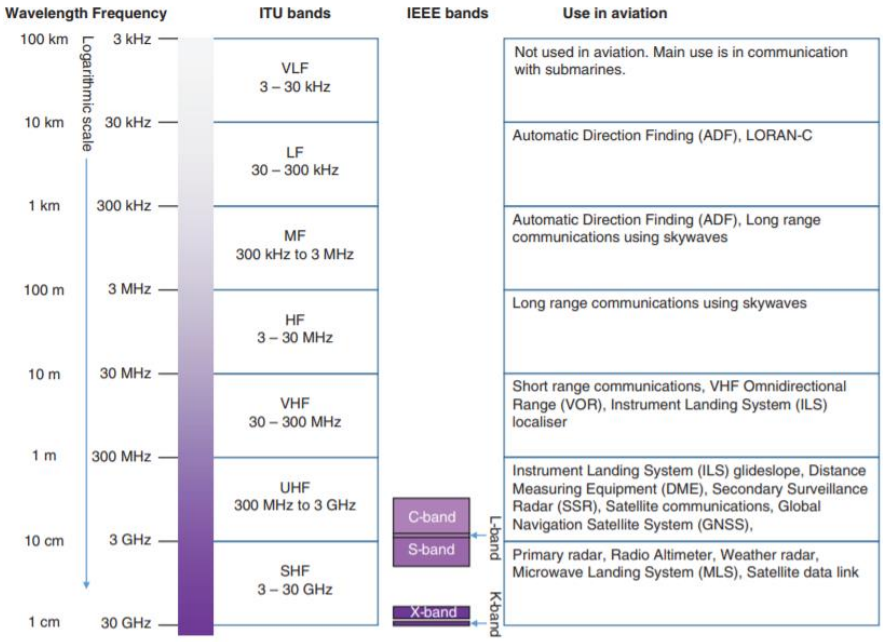

Figura 1.8: distribución de espectro**.**

CAPÍTULO 2: GENERALIDADES DE LAS AERONAVES

2.1 Introducción

Una aeronave es cualquier vehículo capaz de navegar por el aire o por la atmosfera de un planeta. Según la AOCI una aeronave es "toda máquina que puede desplazarse en la atmosfera por reacciones del aire que no sean las reacciones del mismo contra la superficie de la tierra". En el caso de las aeronaves tipo ala fija, hay 4 fuerzas que actúan sobre esta durante el vuelo: resistencia, empuje o tracción, sustentación y peso tal como lo muestra la figura 2.1:

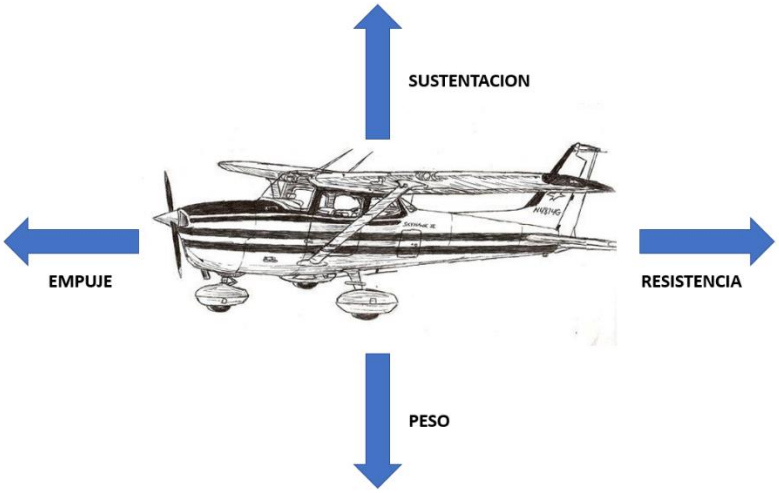

Figura 2.1 fuerzas presentes en una aeronave.

Estas fuerzas actúan en pares, la resistencia es opuesta a la tracción y la sustentación es opuesta al peso. En el momento en que la aeronave se mantiene en vuelo la tracción es igual al peso y la resistencia es igual a la tracción, de esta manera la fuerza neta es equivalente a cero. La resistencia es la fuerza que actúa en dirección contraria al movimiento de un objeto en el fluido que lo rodea, en este caso el aire. La energía utilizada para impulsar una aeronave a través del aire genera una resistencia que disminuye la velocidad. El empuje o tracción es la fuerza que hace que el avión avance y contrarreste la resistencia del aire.

La parte principal de una aeronave son las alas, ya que producen la fuerza de sustentación que le permite volar a la aeronave. Para ello se diseña con un perfil aerodinámico especial llamado airfoil, la parte superior es llamada extradós y la parte interior intradós, tal como la muestra la figura 2.2. El aire al hacer contacto con el borde de ataque, este desvía una cantidad al extradós e intradós, ambas masas de aire salen por el borde fuga a la misma velocidad debido a que en el extradós el aire circula con una velocidad y presión mayor que en la parte de abajo, lo que permite que las alas generen la fuerza de sustentación que elevan el avión durante el despegue y lo mantiene a flote durante el vuelo. En el aire, la fuerza neta es cero porque la sustentación es igual al peso del avión, que incluye a la gravedad. [5]

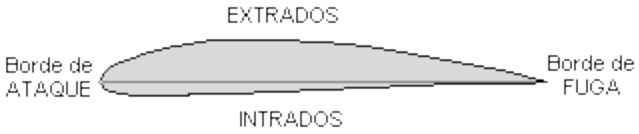

Figura 2.2 Perfil alar.

En la cola de la aeronave está el estabilizador horizontal y vertical. Son elementos que aseguran la estabilidad del avión; es decir, su tendencia a regresar a un estado inicial tras una perturbación. El horizontal se encarga de estabilizar los movimientos de arriba a abajo de la cabeza de la aeronave con respecto a un eje lateral, mientras que el estabilizador vertical los movimientos de izquierda a derecha respecto al eje vertical tal como lo muestra la figura 2.3.

Figura 2.3 Estabilizador vertical y horizontal.

Los mandos de vuelo son los mecanismos que permiten cambiar la orientación y la posición de la aeronave. Las tres superficies de mando principales son:

- Elevadores: están en la parte trasera de la aeronave, permiten que esta ascienda o descienda y genera un movimiento que se conoce como cabeceo.
- Timón de dirección: está en la cola y hace que la cabeza de la aeronave gire hacia la izquierda o la derecha, lo que se conoce como la maniobra de guiñada. Funciona como el timón de un barco: un giro de la superficie hacia la derecha cambia de dirección a la derecha.

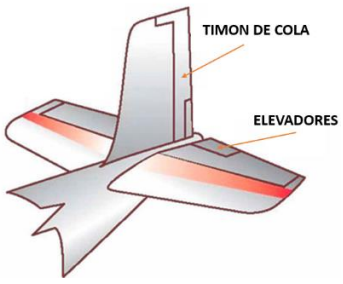

Figura 2.4 Superficies de mando.

- Alerones: están en las alas y se activan en sentidos opuestos para que el avión se incline de un lado hacía otro también conocido como alabeo. Si el piloto quiere inclinar el avión hacia la izquierda, tendría que flexionar el alerón izquierdo hacia arriba y el del ala derecha hacia abajo, tal como lo muestra la figura 2.5. [6]

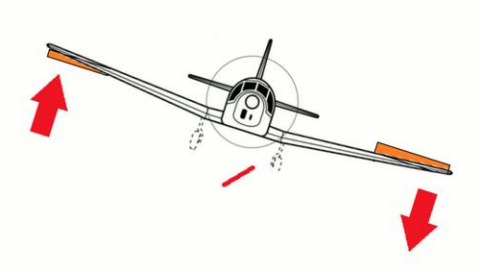

Figura 2.5 Alerones en una aeronave.

2.2 Aviación privada.

La aviación privada se entiende la realizada por personas naturales o jurídicas con fines distintos a los comerciales, tales como deportivos, recreativos, o como elemento complementario de otras actividades industriales o comerciales como la aviación ejecutiva. Se entiende que la aviación privada – ejecutiva no tiene como propósito el generar ingresos para el piloto, también lo es, que la misma es motor decisivo para la generación de riqueza y desarrollo socio-económico para las pequeñas y grandes ciudades (cumbres internacionales, ruedas de negocios, desarrollo del objeto social de las empresas, inversión extranjera, entre otras). Sin embargo, esta modalidad de aviación se enfrenta diariamente a múltiples desafíos locales los cuales deben ser superados con el fin de crear estándares y mejores prácticas en una industria tan dinámica y creciente como lo es la de la aviación en Colombia.

Para ser piloto privado se debe tener la licencia PPL (piloto privado de avión). Esta formación suele durar de 6 a 9 meses y consta de unas 100 horas lectivas y 45 horas de vuelo, en total se estudian alrededor de 9 asignaturas que comprenden conocimientos sobre aeronaves, meteorología, legislación aérea entre otras cosas. El piloto al final saldrá con la licencia de vuelo para operar aeronaves livianas monomotor (Cessna 172) o bimotor (Diamond DA 42) dependiendo de los estudios realizados. [7]

2.3 Aviación comercial.

La aviación de transporte comercial es una actividad que hacen las compañías aéreas, ya sean éstas grandes o pequeñas, dedicadas al transporte aéreo de personas o mercancías con una cierta frecuencia en sus rutas, se utilizan por lo general aeronaves de tipo comerciales, generalmente grandes, usadas para el trasporte de pasajeros. Estas aeronaves están operadas por aerolíneas. Aunque la definición puede cambiar de país a país, una aeronave de línea se define típicamente como un avión para transportar múltiples pasajeros o carga en servicio comercial.

Figura 2.6 Boening 747.

Las aeronaves comerciales con mayor capacidad son los aviones de fuselaje ancho. Estas aeronaves generalmente contienen un doble pasillo que recorre todo el aparato desde la cabeza hasta la cola. El uso de este tipo de aeronaves esta normalmente reservado a vuelos de larga distancia entre centros de conexión. Las aeronaves más comunes son los de fuselaje estrecho o aviones de único pasillo. El uso más común de estos aviones es el de vuelos de corta-media distancia.

Figura 2.7 Boening 737.

Un piloto comercial además de tener la licencia PPL, es necesario que tenga también la licencia CPL (piloto comercial) y como mínimo 150 horas de vuelo. Esta licencia capacita para ser copiloto en transporte aéreo comercial y se compone de un módulo de 15 horas. Después de las 200 horas de vuelo ya se podrá trabajar en una aerolínea como primer oficial al mando y al completar más de 1500 horas de vuelo el piloto podrá tendrá la experiencia y conocimiento para ser capitán de la aeronave. Existen otros módulos que capacitan al piloto para trabajar bajo ciertos factores, tales como vuelos nocturnos o vuelo por instrumentos.

Al ser una responsabilidad tan grande manejar estos aparatos antes y después de volarlos se deben hacer ciertas cantidades de horas en simulador para afianzar conocimientos y habilidades, es aquí la importancia que un simulador sea lo más real a una cabina de una aeronave con todas sus funcionalidades y entre más plus se le agreguen mejor preparados saldrán los pilotos entrenados. Cabe aclarar que ya un piloto graduado con un número de horas de vuelo, debe seguir en constante entrenamiento en simulador exigido por la aerolínea por la que opere.

2.4 Simuladores de vuelo (estructuras).

Un simulador de vuelo es un sistema que intenta replicar o simular la experiencia de pilotear una aeronave de la forma más precisa y real posible. Los diferentes tipos de simuladores de vuelo van desde videojuegos, consolas equipadas con diferentes actuadores e indicadores (conexión vía USB) hasta réplicas de cabinas en tamaño real montadas en accionadores hidráulicos o electromecánicos controlados por sistemas modernos computarizados, construidos por las fábricas que diseña y construyen aeronaves: AIRBUS y BOENING.

Figura 2.8 Simuladores Airbus a320, Tolouse

Los simuladores de vuelo del centro de pruebas de Airbus (Saint-Martin, Tolouse), se mantienen en funcionamiento durante al menos 11 horas diarias. Cuenta con los últimos sistemas certificados para una aeronave y son utilizados por los pilotos de las aerolíneas para familiarizarse con un tipo de avión de la compañía. [8]

Hoy en día hay varias categorías de simuladores de vuelo utilizados para el entrenamiento de pilotos. Las mismas que van desde simples sistemas de entrenamiento básico hasta simuladores de vuelo con 6 ángulos de movimiento, que son denominados sistemas complejos. Estos simuladores, de última generación, al igual que los simuladores simples, básicamente son utilizados para el entrenamiento de pilotos, cuya función esencial es la de capacitar a la tripulación en procedimientos normales, anormales y de emergencia, antes y durante el vuelo, practicando innumerables situaciones, tales como: fallas en los sistemas electrónicos, perdidas de potencia, vientos de cola y muchos otros, que no pueden ser realizados de forma segura con una aeronave en situaciones reales. Los simuladores son evaluados por instituciones gubernamentales tales como la Administración Federal de Aviación de Estados Unidos (FAA) y Direcciones de Aeronáutica Civil de diferentes países, las cuales clasifican, regulan y certifican estos dispositivos según su categoría en niveles A (simulador con la interfaz), B (simulador de la cabina), C (simulador full motion) y D (simulador full motion con interacción externa). La principal exigencia para la certificación de estos equipos consiste en demostrar que sus características de vuelo coinciden exactamente con las de la aeronave para la cual fue fabricada el simulador.

Existen simuladores denominados Full Motion Flight Simulator, donde se replica los aspectos de una aeronave específica y su entorno. Este tipo de simuladores pueden generar movimiento de modo en que los ocupantes sientan un nivel de realismo tal como sucedería en una aeronave real, engañando a las tripulaciones y haciéndoles creer que se encuentran volando. Adicionalmente estos simuladores cuentan con funcionalidades para instructores, sensaciones dentro de la cabina, donde el instructor puede rápidamente crear cualquier situación anormal o de emergencia que puede ir desde: fuego en los motores, mal funcionamiento del tren de aterrizaje, fallas eléctricas, tormentas, riego de colisiones con otras aeronaves, fallas en los sistemas de navegación, etc. Pero existe un problema de acceso

a estos simuladores, el costo, que para escuelas pequeñas de aviación se hace imposible pagar, alrededor de 250.000 a 500.000 dólares por cada uno de ellos. [9]
Esto ha impulsado que la comunidad Maker haya podido realiza estructuras, módulos y planos a precios bajos colgados en la web con el fin de que cualquier persona tenga acceso a estos, prácticamente se compone de una estructura en madera en la cual van ubicados cierta cantidad de periféricos incluidos actuadores e indicadores que son controlados por un microcontrolador, y, este a su vez, está conectado con un simulador de tipo software recibiendo y mandando información, por otro lado, los instrumentos de vuelo se simulan en pantallas con datos sacados en tiempo real del simulador de vuelo.

2.5 Simuladores de Vuelo (Software).

Hay diversos tipos de simuladores de vuelo. Como grandes grupos se pueden distinguir los simuladores para ingeniería y los simuladores para entrenamiento. Los primeros se usan para desarrollar la aeronave, las leyes de control, reducir horas de ensayos en vuelo, investigación, etc. Un simulador de vuelo, como mínimo, suele estar compuesto por: un modelo aerodinámico donde se caracterizan las fuerzas y momentos aerodinámicos, modelo dinámico donde se incluyen las ecuaciones diferenciales que relacionan la posición, velocidad y actitud con las deflexiones de superficies y mando de motor, modelo másico y de inercia, propulsivo donde se relaciona la fuerzas y momentos producidos por el motor, un modelo de controlador de vuelo o flight control que calcula las deflexiones y mando de motor necesarios para realizar las maniobras especificadas, otro de estimación y navegación donde se estiman posición, velocidad y actitud a partir de medidas ruidosas y modelos de actuadores y de sensores. Para cumplir con su funcionalidad todos estos modelos deben ser muy fieles al sistema real que se quiere simular. Normalmente, en este tipo de simuladores, se requiere menos fidelidad en la interfaz hombre-máquina (cabina) y en el sistema de presentación visual (mundo).

Dentro de los simuladores de ingeniería, hay dos tipos claramente diferenciados: los simuladores que corren “off line” y los denominados “software in the loop”. Los simuladores “off line”, son los más simples, pues no corren en tiempo real y pueden ejecutarse en un ordenador sin necesidad de un RTOS (sistema operativo de tiempo real). Sirven para desarrollar leyes de control, navegación e investigación. Por otro lado, los simuladores “software in the loop”, corren un RTOS, y todo se simula por software (incluidos sensores y actuadores). Pueden estar formados por varios ordenadores, donde en uno se corre el modelo 3D y en otro se ejecuta el modelo dinámico de navegación y control. Dentro de los simuladores software-in-the-loop, se pueden distinguir cuatro tipos: sistemas aéreos no tripulados también conocidos como RPAS (remote pilot aereal systems), aeronaves tripuladas, simuladores hardware-in-the-loop y los denominados in-flight simulators.

Un simulador de sistemas aéreos no tripulados consiste en simular el modelo de la aeronave con todos sus modelos matemáticos y físicos controlados por una Ground Control Station (GCS), se simula en 3D la imagen de las cámaras al apuntar sobre un terreno virtual. Por otro lado, los simuladores de aeronaves tripuladas suelen consistir en un programa de ordenador como es el caso de Flight Simulator X, X-plane 11, Repard 3D. Estos simuladores carecen de una GCS y poseen un módulo de gestión de vuelo, en vez de simulación de una cámara hay una representación 3D del mundo real y una representación virtual de la cabina que

suelen ser funcionalmente similar a las pantallas que el piloto vería en el modelo de la aeronave real. Los simuladores hardware-in-the-loop, corren en tiempo real con RTOS y el sistema de control de vuelo no se simula, si no que las entradas son señales físicas (normalmente eléctricas), sintetizadas en un ordenador o generadas por sensores reales alimentados por magnitudes físicas generadas por dispositivos especiales. Por último, los simuladores in-flight simulators se utilizan en la investigación de las llamadas Handling Qualities o Cualidades de Vuelo. Se trata de aeronaves reales y la simulación se realiza durante el vuelo de forma que se comporte como otra aeronave distinta.

En cuanto a la cabina, simplemente decir que debe replicar en mayor o menor medida la propia cabina de la aeronave, con todos sus sistemas. El grado de realismo, en este caso, no depende tanto de la funcionalidad puesto que todos van encaminados al entrenamiento, si no que tiene que ver con la categoría dentro de la cual se quiere certificar el simulador.

2.6 Instrumentación a Bordo

Los instrumentos básicos que tiene una aeronave de entrenamiento para un piloto en formación y que han sido recreados para este simulador son los siguientes:

Figura 2.9 Anemómetro

Figura 2.10 Indicador de actitud

Figura 2.11 Altímetro

En la figura 2.9 se observa el indicador de “velocidad de viento” o también conocido como “anemómetro”, es un instrumento que mide la velocidad relativa de la aeronave con respecto al aire que se mueve, por lo general la unidad de medida es m.p.h (millas por hora), nudos o ambas unidades. Además de ser el instrumento más importante para un piloto, tiene una marca de colores ya estandarizado que permite a este determinar a simple vista las limitaciones en las velocidades. La franja blanca indica el rango de operación con flaps, la franja verde indica el rango de operación normal, la franja amarilla indica el rango de operación con posibles riesgos en la estructura de la aeronave y a franja roja indica que nunca se debe exceder la velocidad hasta ese punto.

En la figura 2.10 se observa el “indicador de actitud” o también llamado “horizonte artificial”, es un instrumento que proporciona al piloto una referencia inmediata si la aeronave esta inclinada lateralmente, el morro (nariz del fuselaje) hacia arriba o abajo con respecto al horizonte. En condiciones de visibilidad reducida o nula este instrumento es el que permite la navegación de la aeronave en dicho espacio.

En la figura 2.11 se observa el "altímetro", este indicador muestra la altura a la cual está volando la aeronave, la unidad de medida son Pies, funciona bajo principios de presión y temperatura. Por lo general esta graduada con números que van desde 0 a 9, consta de 2 agujas, la menor indica miles de pies y la mayor centena de pies, se debe leer de menor a mayor, una aeronave por debajo de 10.000 pies es visible. [10]

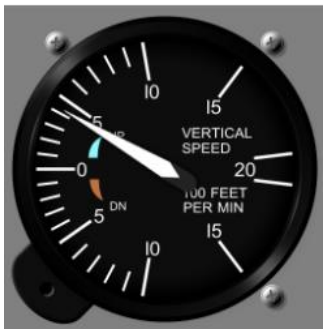

Figura 2.12 Indicador de viraje Figura 2.13 Indicador de dirección Figura 2.14 variometro

En la figura 2.12 se observan 2 instrumentos independientes en la misma figura: el "indicador de viraje" (perfil de un avión en miniatura) y el "indicador de coordinación de viraje" (bola dentro de un tubo). El indicador de viraje muestra si la aeronave está girando, hacía que lado lo hace y cuál es la velocidad angular con la que realiza el giro, cuando las alas de la aeronave en miniatura se alinean con cual quiera de las 2 marcas L o R, tendrá una velocidad angular de viraje o giro estándar. Por otra parte, el indicador de coordinación de viraje indica que la aeronave está derrapando (cuando la "bola" cae contrario al lado del viraje) o está resbalando (cuando la "bola" cae hacia el lado del viraje). Es importante para el piloto que la bola siempre debe mantenerse centrada en todo momento tanto en los giros como en vuelos rectos, a esto se le conoce como calidad de giro.

En la figura 2.13 se observa el "indicador de dirección", este instrumento proporciona al piloto una referencia de la dirección de la aeronave, facilitando el control y mantener el rumbo. El desplazamiento de un punto a otro de una aeronave se realiza a través de una ruta, esta ruta está compuesta de uno o más tramos, los cuales para navegar de un punto al siguiente se debe seguir una determinada dirección o rumbo. Este instrumento esta graduado en incrementos de 5 grados, con números cada 30 grados y los puntos cardinales indicados por **N** (norte), **S** (sur), **E** (este) y **W** (oeste).

En la figura 2.14 se observa el "indicador de velocidad vertical" o también conocido como "variometro", este instrumento muestra si el avión está ascendiendo, descendiendo o esta nivelado. Además, proporciona la velocidad vertical en pies por minuto (f.p.m) del ascenso o descenso. Este instrumento tiene una única aguja que comienza en 0, las marcas por encima de este indican ascenso, por debajo descenso y en el cero vuelo nivelado. [11]

Figura 2.15 Indicador de revoluciones Figura 2.16 VOR NAV 1 Figura 2.17 VOR NAV 2

En la figura 2.15 se observa el "indicador de revoluciones por minuto (r.p.m)" o también conocido como tacómetro, indica el número de revoluciones por minuto en el motor (está directamente relacionado con el cigüeñal) y es la base de referencia para el ajuste de la potencia, ya que el consumo de gasolina puede verse comprometido por un número excesivo de RPM. El rango verde, es el rango de operación normal que debe estar entre los 2100 y 2700 RPM y el rango rojo so las RMP máximas utilizables (2700 RPM).

En la figura 2.16 se observa el "sistema de aterrizaje instrumental" o también conocido como "VOR Nav1", este dispositivo está basado en señales de muy alta frecuencia (VHF) y por tanto es una radio ayuda para la navegación en ruta de corto alcance, permite guiar al piloto en las aproximaciones y aterrizajes. El sistema VOR determina el Angulo θ entre la dirección que une la aeronave con la estación VOR y la dirección del norte magnético. Compuesto por dos indicadores del instrumento (agujas blancas) permiten guiar la aeronave en alineación con la pista y descenso hacia la pista.

En la figura 2.17 se observa el "indicador omnidireccional" o también conocido como "VOR Nav2 o Secundario", el principio de funcionamiento es similar al "sistema de aterrizaje instrumental, instrumentos encargados de sintonizar las transmisiones de tierra, permite al piloto saber si se encuentra a la derecha, izquierda o centrado sobre el radial o rumbo (posición) a partir de señales o datos transmitidos por una o varias emisoras específicas. Compuesto por una marcación de rumbo magnético y una aguja indicadora de dirección hacia la emisora VOR, la aguja se desplaza hacia la derecha o izquierda (de manera pendular), indicando la dirección a seguir al rumbo seleccionado.

Figura 2.18 ADF Figura 2.19 Cantidad de combustible Figura 2.20 Temp. de gases

En la figura 2.18 se observa el "ADF" o también conocido como "buscador de dirección automático" es un instrumento de navegación utilizado por lo general en aeronaves ligeras, se basa en captar la máxima intensidad de una señal de baja frecuencia y gran alcance de una emisora NDB (Non-Directional Beacon) en tierra, la aguja del instrumento indica la dirección a dicha estación, esto se utiliza para diferentes propósitos: fijar la posición de la aeronave, navegación en ruta, aproximación por instrumentos, indicar el punto de inicio en un procedimiento de aproximación y espera en procedimientos.

En la figura 2.19 se observa el “indicador de cantidad de combustible”, muestra la cantidad de combustible existente del ala izquierda (LEFT) y derecha (RIGHT), la capacidad máxima en la aeronave Cessna 172 es de 26 galones.

En la figura 2.20 se observa 2 instrumentos dentro del indicador, el de la izquierda que tiene 3 siglas EGT es el “indicador de temperatura de salida de gases”, se utiliza para conseguir la mejor y optima mezcla de combustible-aire. Para lograr esta eficiencia se aumenta la potencia del motor hasta alcanzar el máximo EGT permitido y con la palanca de mezcla se baja la temperatura hasta 37° Celsius, obteniendo la potencial ideal. La mezcla más económica para el vuelo se consigo alrededor de los -3° hasta 10° Celsius. Del lado derecho está el “indicador del flujo de combustible”, que muestra la cantidad de combustible que suministra al motor en galones por hora, para conseguir regímenes económicos, hay que mantener la aguja en el sector señalado en verde.

Figura 2.21 manómetro de succión

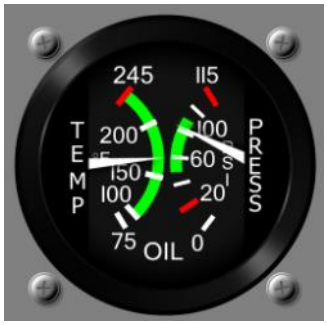

Figura 2.22 Indicador de temperatura

En la figura 2.21 se observa 2 instrumentos en el indicador, del lado izquierdo se encuentra el “indicador del manómetro de succión”, indica si la bomba de vacío funciona correctamente. Los valores normales de funcionamiento se encuentran en el sector verde: 4,5 y 5,5 pulgadas de mercurio. Si se avería la bomba de vacío, los giroscopios del “indicador de actitud y dirección” no funcionaran correctamente. Del lado derecho se encuentra el “amperímetro” indica si el generador o alternador está cargando la batería, si la aguja es por encima de 0, quiere decir que la batería está siendo cargada por el alternador, de lo contrario si la aguja está por debajo de 0, la aeronave está consumiendo más energía eléctrica de la que el alternador está suministrando, por lo que se procede a desconectar algún sistema que no resulte imprescindible y que consuma energía, la medición se realiza en amperios.

En la figura 2.22 se observa en el indicador 2 instrumentos, del lado izquierdo se encuentra el “indicador de temperatura”, este muestra la temperatura del aceite del motor. El intervalo normal de funcionamiento está representado por el sector de color verde, si la temperatura haciende por encima de los valores normales, se debe aterrizar inmediatamente para evitar daños en el motor. Del lado derecho se encuentra el “indicador de presión de aceite”, indica si el aceite fluye a través del motor, el intervalo de normal de funcionamiento está representado por el sector de color verde. A baja presión se debe aterrizar de inmediato y con alta presión no es tan grave, pero puede generar fugas en el sistema o producir exceso de consumo. [12]

2.7 Switchs y controles a Bordo

Los actuadores básicos que se encuentran en una aeronave de entrenamiento para un piloto en formación y que han sido recreados para este simulador son los siguientes:

Figura 2.23 módulo de botones y switches

La figura 2.23 se observa el grupo de luces y electrónica de la aeronave, cada botón esta numerado con el fin de describir su funcionamiento:

1. Alternador: genera energía para cargar la batería y soporta la alimentación de los equipos eléctricos.
2. Batería: Proporciona electricidad a la aeronave.
3. Bomba de combustible: Permite el paso de combustible del tanque al motor, cuando la aeronave se encuentra apagada, se debe encender por 5 segundos para permitir el flujo de combustible.
4. Luz de rotación del radiografo o Beacon: es una luz roja de alta intensidad ubicadas por lo general en la parte superior e inferior de la aeronave, parpadean en intervalos regulares, funcionan como un rotativo similar al de una ambulancia. Hacen parte del sistema de anticolisión de la aeronave, está activa tanto de día como de noche, además provee de información al personal de tierra de que los motores están encendidos o a punto de encender
5. Luces de aterrizaje: son las luces más potentes que posee una aeronave, de color blanco y situadas generalmente en alas y cola, utilizadas para iluminar el terreno y la pista durante el despegue y el aterrizaje.
6. Luces de taxi: se utilizan para iluminar la pista que está en frente, desde el momento en que se empieza a moverse la aeronave hasta que despega.
7. Luces de navegación: las luces de navegación indican la posición relativa de la aeronave, se utilizan entre el ocaso y salida del sol y también en tierra en todo momento. El esquema con el que están ubicadas es, una luz roja a la izquierda y una luz verde hacia la derecha en sentido avante (adelante) más una blanca en la cola. Las 3 luces cubren un arco de 110°
8. Luces estroboscópicas: forman parte de un sistema de anticolisión para evitar accidentes en vuelo con otras aeronaves, ver y ser visto en condiciones de baja visibilidad, se utilizan entre el ocaso y la salida del sol, también cuando hay nubes y niebla.

9. Tubo pitot: el sistema pitot y estática se encarga de proporcionar las presiones a los instrumentos que depende de este (Indicador de velocidad vertical), su función es captar el flujo del aire de impacto que tiene la aeronave contra la masa de aire a medir, si bien no es un elemento principal del sistema, el tubo pitot incorpora una resistencia eléctrica que puede ser conectada en caso de esta bloqueada por hielo.
10. Aviónica: Proporciona electricidad a la aviónica de la aeronave, esta abarca el grupo de comunicación (Comunicación, Navegación, transponder y piloto automático) y los indicadores de VOR1, VO2 y ADF.

En la Figura 2.24 se puede observar la distribución de los 5 tipos de luces en una aeronave Cessna 172:

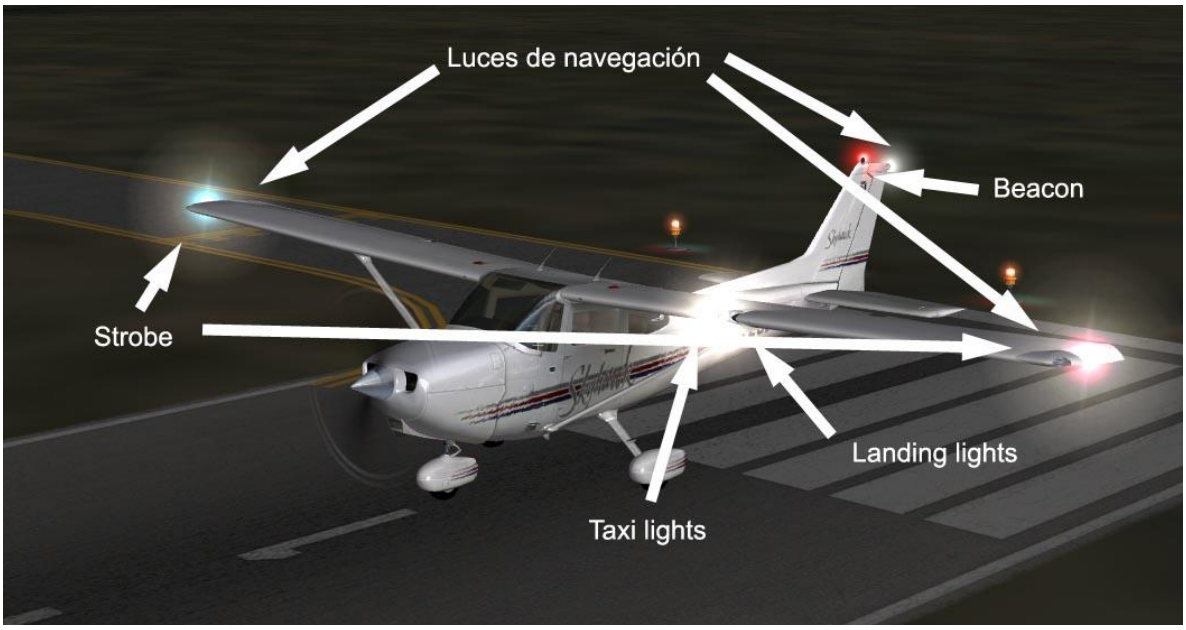

Figura 2.24 Posicionamiento de luces en una aeronave

La figura 2.25 se observa el grupo de potencia y flaps de la aeronave, cada botón esta numerado con el fin de describir su funcionamiento:

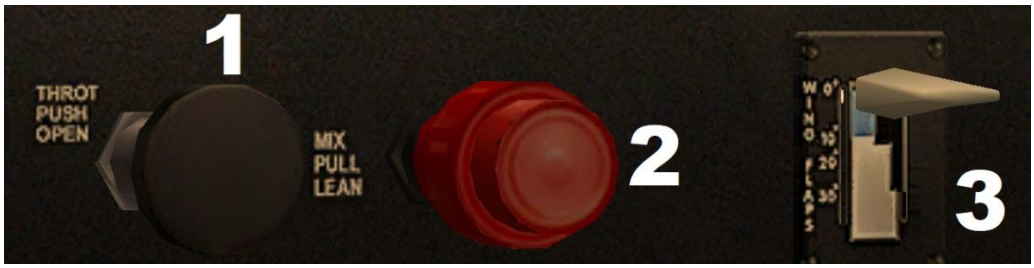

Figura 2.25 módulo de potencia

1. Palanca de aceleración: es una palanca que se encuentra en la cabina cerca de la consola central de la aeronave y usada por el piloto o copiloto para controlar el empuje de los motores. Al accionar la palanca hacia el frente, los interruptores abren las

válvulas de combustible, que permanece abierta durante el funcionamiento normal de la aeronave en el despegue y vuelo.

2. Palanca de mezcla: este tipo de aeronaves (ligeras) funcionan quemando una mezcla de X partes de aire por X partes de combustible, a esta relación se le denomina mezcla. Esta palanca mezcla el combustible con el aire del ambiente, ya que a medida que la aeronave aumenta de altitud, disminuye el oxígeno y por lo tanto hay que disminuir el combustible para que haya una mejor combustión, de esta manera aumenta la distancia de vuelo.
3. Flaps: es un sistema diseñado para aumentar la sustentación (fuerza generada sobre un cuerpo que se desplaza a través de un fluido) en determinadas fases del vuelo de una aeronave, con el fin de aumentar la cuerda aerodinámica y a curvatura del perfil alar, de tal modo que la velocidad en el aterrizaje o despegue se reduzca significativamente permitiendo un vuelo más lento que en el crucero.

El equipo de comunicación y navegación, transpondedor y piloto automático perecen al grupo de comunicaciones de la aeronave, en la figura 2.26 se observa los indicadores y actuadores de la comunicación y navegación, cada botón esta numerado con el fin de describir su funcionamiento:

Figura 2.26 módulo de comunicaciones.

1. Display Comm1: es un indicador de frecuencia de comunicación que sintoniza el piloto para tener contacto con la torre control que cubre la zona por la cual está navegando la aeronave. Del lado izquierdo del indicador esta la frecuencia principal (actual) y del lado derecho la frecuencia (stanby) secundaria que el piloto sintoniza para tenerla lista.
2. Selector de frecuencia Comm1: es un conjunto de encoders rotativo que ajustan la frecuencia secundaria de comunicación, el exterior define los Mhz de uno en uno y el interior los Khz de 50 en 50.
3. Display Nav1: es un indicador de frecuencia de navegación que sintoniza el piloto para saber hacia dónde debe o desea volar, esta frecuencia la emiten pequeñas estaciones ubicadas en puntos fijos y aeropuertos.
4. Selector de frecuencia Nav1: es un conjunto de encoders rotativo que ajustan la frecuencia secundaria de navegación, el exterior define los Mhz de uno en uno y el interior los Khz de 50 en 50.
5. Pulsador de navegación: activa la frecuencia de navegación.

El transpondedor (figura 2.27) es un dispositivo electrónico que produce una respuesta cuando recibe una llamada de radio frecuencia, además facilitar la identificación de la aeronave en el control de tráfico aéreo. Los códigos que emite el transpondedor son números de 4 dígitos, y son utilizados a nivel global, por ejemplo 7500 significa que la aeronave está secuestrada, 7600 comunicación perdida, 7700 emergencia general.

Figura 2.27 módulo de transponder

1. Indicador de código del transpondedor.
2. Interruptores que van con numeración de 0 a 7 para seleccionar el código necesario.

El piloto automático (figura 2.28) está compuesto por 2 indicadores y 8 interruptores, regula de manera automática la altura y el rumbo que lleva la aeronave.

Figura 2.28 módulo de piloto automático.

1. ALT: Indicador de altura (pies).
2. VS: Indicador de velocidad vertical (pies/min).
3. UP: aumenta la altura (pies) de 100 en 100.
4. DN: disminuye la altura de (pies) 100 en 100.
5. AP: Interruptor de activación del piloto automático.
6. HDG: Interruptor de activación de rumbo preconfigurado por el indicador de dirección.
7. NAV: permite seguir la radial fijada por el VOR1.
8. APR: permite activar el seguimiento de la señal del sistema de aterrizaje instrumental.
9. REV: permite alejar la aeronave de una pista y del aeropuerto, se sintoniza en NAV1 la frecuencia de la pista que se deja atrás.
10. ALT: activa la altura preconfigurada. [13]

CAPÍTULO 3: DISEÑOS Y COMPONENTES.

3.1 Estructura de un simulador de vuelo: Concepto.

La idea principal del simulador es poder entregarle a un piloto en formación un sistema real, lo más parecido a una cabina de una aeronave en cuestión de actuadores, botones e indicadores. Existe mucha información y comunidades en la web que dan soporte y guías de cómo construir un simulador de vuelo casero de bajo costo (low cost) paso a paso con sistemas electrónicos de uso cotidiano y tarjetas de desarrollo de programación básica tal como es Arduino [14]. El propósito de este simulador en una versión final, es ser avalado al menos por la Aerocivil (entidad administrativa colombiana especial en aeronáutica civil) o con mayor rigurosidad por la FAA (administración federal de aviación) para instrucción en centros de entrenamiento. Para ello, la tarjeta de desarrollo a implementar debe ser capaz de hacer procesos de manera paralela, cálculos rigurosos y sin que haya desborde del sistema de memoria por tanto tráfico de datos, más detalladamente un sistema operativo que se encargue de administrar todos estos procesos pero que tenga protocolos y pines de comunicación, esto definirá la robustez del simulador. Para construir el simulador se debe primero establecer qué cantidad de módulos se van a implementar para proceder a su respectivo acople y construcción de acuerdo a si es una cabina de aeronave pequeña o grande.

3.1.1 Modulo De Software.

Figura 3.1 modulo PC - Software

Este primer módulo, comprende el equipo, características necesarias para ejecutar el software del simulador y los plugins necesarios para extraer la información y los cambios del simulador a un segundo plano. En este caso, el simulador con el cual se trabaja es el Microsoft flight simulator X, basado en arquitectura X86, no necesita GPU ni potencia de computo para renderizar escenarios, basta con los procesadores que actualmente Intel tiene en producción mayores a 1,0 GHz y 512Mbs de RAM para un buen funcionamiento [15]. De manera simultánea, para la extracción de datos a un segundo plano, existen muchos plugins gratuitos,

de pago y fáciles de usar pero que no dan libertad o autonomía para un óptimo funcionamiento. Hace algunos años Microsoft desarrollo un SDK (Visual Basic y C#) para diseñar o modificar modulos y componentes en el fligth simulator, el único problema es que el SDK ya no tiene soporte por parte de la empresa. Existe un demo creado con este SDK por una empresa neozelandesa, link2fs, cuyo tipo de licencia es GNU, además de ser gratuita, permite acceso a casi un 90% de opciones en cabina de cualquier aeronave, para el caso del Cessna 172, está al 100% en las funcionalidades, así que este será el plugin o extensión a utilizar dentro del sistema completo del simulador de vuelo.

Figura 3.2 modulo PC – Software - Interfaz

3.1.2 Modulo de Hardware.

Del módulo anterior, se tiene la información necesaria para procesar y visualizar cada cambio que haya en el simulador, ahora es necesario de una conexión hardware para poder conectar los actuadores e indicadores y así crear la interfaz que retroalimente la información tanto de recepción como de transmisión. Para ello, se utiliza una tarjeta de desarrollo que en lo posible tenga un sistema operativo que controle en tiempo real cada tarea, y disponga de diferentes protocolos de comunicación para poder conectarse a cualquier dispositivo. Además, dicha tarjeta al menos debe garantizar conexión de tipo ethernet con el fin de poder conectarse a través de ella desde cualquier punto de la red. Para este simulador, se hará uso de la tarjeta Beaglebone black por características de procesamiento y amplia disponibilidad de pines. Por otro lado, está la opción de un microcontrolador de la familiar ARM, más exactamente una Stm32f4, en caso de necesitar más pines para periféricos.

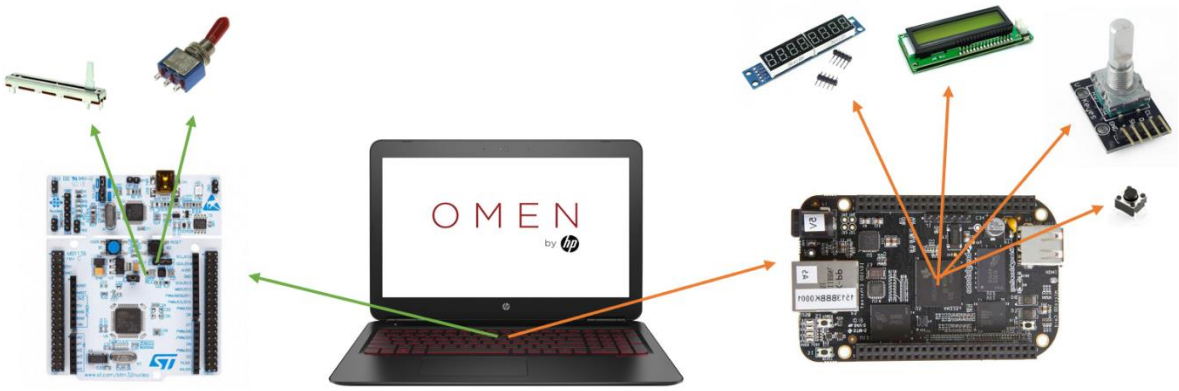

Figura 3.3 Modulo PC – Boards – Periféricos.

3.1.3 Modulo de instrumentos.

Tal vez representar los instrumentos de una aeronave sea la parte más complicada de la interfaz del simulador, ya que en muchos casos funcionan estos indicadores con transductores que cambian señales de presión o velocidad a desplazamiento. Por esta razón es preferible trabajarlos de manera digital y acoplarlo en una pantalla LCD dependiendo del tamaño de la cabina. Para ello, se hace uso del software AirManager el cual hace una interfaz virtual con el flight simulator para la extracción de los valores y proceder a graficar cada dato en su respectivo instrumento.

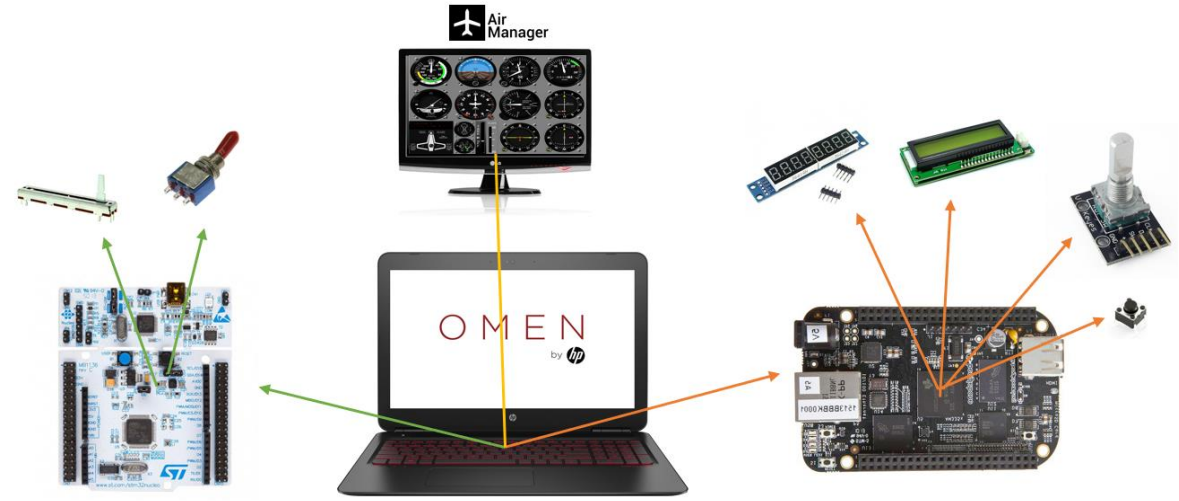

Figura 3.4 Modulo PC – Boards - Instrumentos

Este es el concepto final que hay detrás del simulador para su construcción. Solamente queda definir y organizar los componentes electrónicos implementados en la elaboración de este simulador de vuelo.

3.2 Tarjetas de desarrollo.

Para garantizar un óptimo funcionamiento del simulador, se debe trabajar con alguna board de desarrollo en lo posible de hardware libre y que maneje algún sistema operativo que soporte múltiples tareas al tiempo, que sea de bajo consumo, pero lo más importante es que permita acceso tanto al software como hardware y que cuente con gran cantidad de pines y protocolos de comunicación. En un mundo donde los microprocesadores embebidos están en su mayor auge, aparece la tarjeta de desarrollo beaglebone black soportada por Texas instrument, cuenta con funciones de una computadora básica y cabe en la palma de la mano, el chip principal, AM335x incluye una CPU ARM cortex-a8 (soporta Linux o andorid), un DSP TMS320C64 para decodificación acelerada de audio y video, y una GPU powerVR SGX530 que provee renderizar en 2D y 3D que soporta OpenGL. Cuenta también con salidas de video S-video y HDMI por separadas. Entrada para tarjetas SD/MMC, puerto USB, conector serial RS-282 y conexión JTAG. El almacenamiento interno y la memoria son provistos por un chip que incluye 256 Mb de memoria flash y 512 Mb de RAM, además de las 2 regletas de 46 pines que tiene en cada extremo la tarjeta para cualquier propósito. [16]

Por otro lado, en caso de requerir de más pines se tiene como opción utilizar la board de desarrollo SMT32f4 núcleo, proporciona conceptos de forma asequible y flexible por la gran cantidad de documentación debido a la programación en C donde es fácil construir prototipos combinando rendimiento y consumo energético. Esta placa no requiere de ninguna herramienta hardware externa ya que integra el depurador y programador por medio del STLink/V2-1, este microcontrolador pertenece a la familia cortex-M4 y opera a 85 MHz, cuenta con 512 Kb de flash, claramente para tareas básicas. [17]

3.3 Componentes electrónicos implementados en el simulador de vuelo.

En la implementación de la cabina del simulador de vuelo, se utilizan componentes pasivos y activos, algunos como actuadores que son los dispositivos de manipulación por parte del piloto y otros como indicadores que son los que entregan información directamente del simulador, con el fin de que sea lo más real a una cabina, se da a conocer cada uno de estos componentes y para qué son utilizados.

3.3.1 Modulo display 7 segmentos de 8 dígitos, driver max7219.

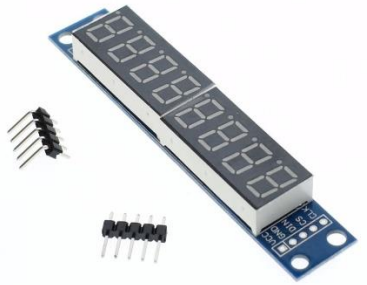

Figura 3.5 Modulo 7 segmentos de 8 displays.

Un visualizador de 7 segmentos es una forma de representar caracteres en equipos electrónicos, compuesto de siete segmentos que se pueden encender o apagar individualmente. Son utilizados dentro de la cabina para representar los valores que muestra

los radios de navegación, comunicación y transpondedor. Este módulo contiene un driver (max7219) para controlar un display de 8 dígitos de 7 segmentos con punto decimal en cada uno de ellos, se comunica a través del protocolo SPI con el microprocesador. Ofrece algunas ventajas utilizar este módulo que gestionar por separado cada despliegue, empezando por las conexiones, que al hacer uso de 5 despliegues solamente para uno de los 4 radios, se debe disponer de más 35 pines en la board. El bus de trabajo funciona a más de 10Mhz, lo que permite registrar cambios en menos de un segundo y generar movimiento de los números en el despliegue, puede encadenar varios dispositivos en modalidad de cascada, cualquier microprocesador de baja potencia puede manipular este módulo sin que haya problemas de parpadeo o redibujado ya que el chip que trae internamente se encarga de manipular la información, lo que único que se debe disponer en la board son 3 pines: MISO, MOSI y CLOCK. [18]

3.3.2 Modulo LCD de 16x2.

Figura 3.6 Despliegue LCD 16x2

El módulo LCD es una pantalla de cristal líquido empleado para la visualización de contenido o información de una forma gráfica mediante caracteres, símbolos o pequeños dibujos dependiendo del modelo. Internamente tiene un chip, que controla su funcionamiento. En este caso, este módulo dispone de 2 filas de 16 caracteres cada una. Dentro del simulador, es utilizado para desplegar toda la información que contiene el piloto automático. Tal vez la pregunta que se genere a partir del uso de ese modulo, es ¿porque razón no se utiliza para los radios de comunicación y navegación? Ya que perfectamente la información de los 4 despliegues de 7 segmentos cabe sin ningún problema en este. La razón por la que solo se utiliza en el piloto automático es que la cabina real está en contacto directo con la luz del sol, los valores de navegación y comunicación son de vital importancia y dicha luz iluminando estas pantallas, quedan completamente verdes, bloqueando la información y generando perdida de comunicación y navegación para un piloto, por esa razón solo es utiliza para el piloto automático. Este módulo tiene 2 métodos de control, por medio de 4 bits o de 8 bits, la única diferencia es la velocidad con la que muestrea la información. Los otros pines son para alimentación y ajuste de brillo de la pantalla. [19]

3.3.3 Encoder rotativos.

Figura 3.7 Encoder rotativo.

Un codificador o encoder rotativo es un elemento que indica mediante una salida codificada su posición, a diferencia de un potenciómetro este encoder no posee topes mecánicos por lo que puede girar libremente. Su funcionamiento es muy sencillo, al girar el eje se genera 2 señales cuadradas, llamadas canal A y B, si el pin común es puesto a masa y cualquiera de los 2 canales a VCC, las señales A y B generadas están desfasadas 90º y su secuencia es: (A,B):(1,0;1,1;0,1) en un sentido de giro, y (A,B):(0,1;1,1;1,0) en el otro, esto se conoce como código de Grey de 2 bits, solamente se debe detectar el flanco de bajada del canal A y ese momento el sentido de rotación viene determinada en el valor de B. En el caso del simulador, estos dispositivos se utilizan para sintonizar la frecuencia tanto de navegación (108.00 – 117.95 Mhz) como de comunicación (118.00 – 136.97 Mhz). [20]

3.3.4 Pulsadores y switches.

Figura 3.8 diferentes pulsadores y switches implementados en una cabina real.

Un pulsador o switch es un operador eléctrico que, cuando se oprime o cambia de posición, permite el paso de la corriente eléctrica y cuando se deja de oprimir o regresa a su estado inicial, lo interrumpe. Por lo general, los contactos de un pulsador están abiertos; es decir, no dejan el paso de corriente. También, existen pulsadores que normalmente tienen los contactos cerrados; es decir, la corriente estará circulando hasta que se oprima. Al pulsar, el circuito se abre y deja de funcionar. Existen pulsadores de muchos tipos y formas, dependen del uso que se va a dar y la corriente que le va circular. Para el caso del simulador se han utilizado los mismos de una cabina real, sin importar su tamaño o función. Solamente en el caso del piloto automático, se ha propuesto un sistema que guarde el estado de los botones en caso de que alguna acción active cualquier de estos botones, se hace por medio de Leds, el botón que se ha presionado, va tener una represión de luz, y en caso de que otro active o desactive dicha funcionalidad, este se va prender o apagar de acuerdo a la lógica que se llegue por programación, ya que el mecanismo de botones no se consigue fácil comercialmente. [21]

3.3.5 Potenciómetro lineal.

Figura 3.9 potenciómetro lineal.

Un potenciómetro es uno de los usos que posee la resistencia o resistor variable mecánica (con cursor y de al menos 3 terminales). El usuario al manipularlo, obtiene entre el terminal central (cursor) y uno de los extremos una fracción de la diferencia de voltaje total, se comporta como un divisor de voltaje. En el caso del potenciómetro lineal, la pista resistiva es recta, de modo que el recorrido del curso también lo es. Son más frágiles que los rotatorios y ocupan más espacio. Además, suelen ser más sensibles al polvo. Para el simulador de vuelo es el transductor ideal para las palancas de potencia y mezcla, ya que están tiene cierto recorrido lineal para cambiar la posición y así ejercer aumento o disminución de la velocidad y consumo de combustible en la aeronave, solamente se debe graduar el desplazamiento y el punto de lectura de dichas palancas para un óptimo funcionamiento. [22]

3.4 Diseño CAD de los componentes electrónicos

Uno de los objetivos secundarios de este proyecto, es realizar un diseño 3D de los componentes para poder mirar su ubicación, construcción y realizar los esquemáticos necesarios para su operación. De esta manera se ha pensado en organizar en 3 módulos todos los componentes electrónicos.

3.4.1 Modulo de comunicaciones.

Este módulo está compuesto por los indicadores y actuadores de navegación, comunicación y piloto automático, tal como se puede apreciar en las siguientes imágenes:

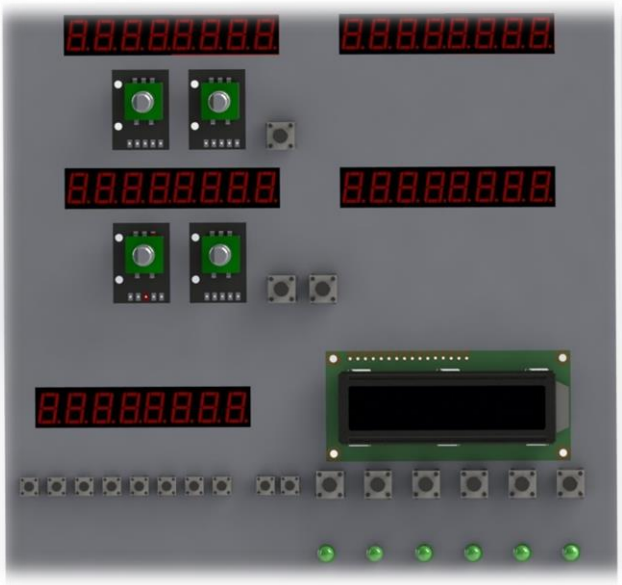

Figura 3.10 Modulo de comunicaciones, transponer y piloto automático (vista de frente).

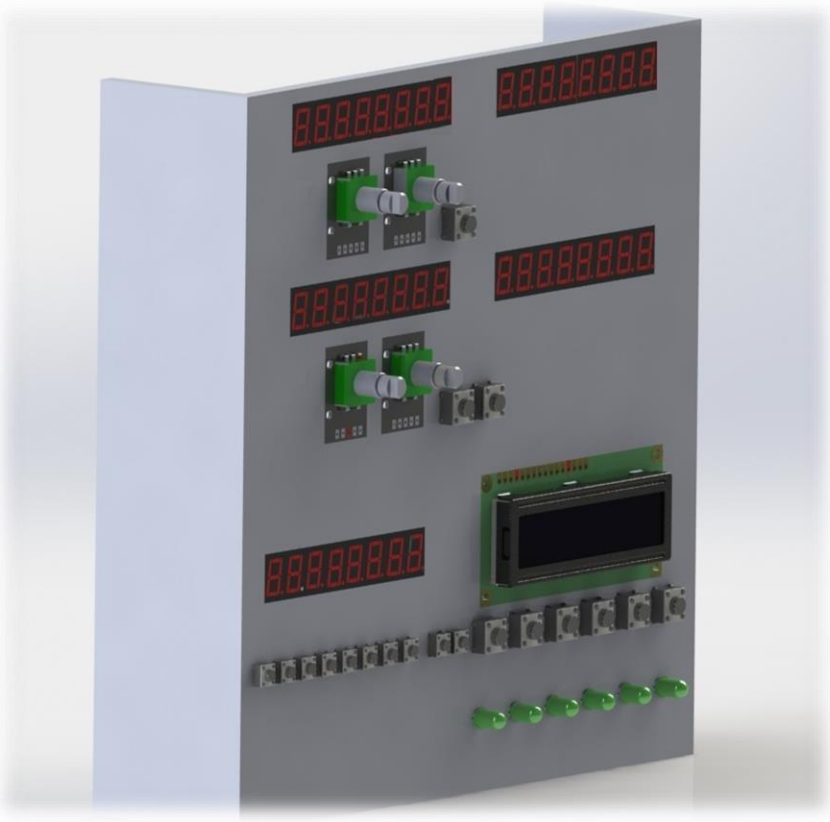

Figura 3.11 Modulo de comunicaciones, transponer y piloto automático (vista lateral).

La idea principal de que estén posicionados los componentes de esta manera es que sea lo más similar a una cabina real, los primeros 2 despliegues pertenecen a los indicadores de comunicación, del lado izquierdo estará la frecuencia principal y del lado derecho la frecuencia stand-by, en la parte de abajo están 2 encoders rotativos que controlan los valores de dichos despliegues tanto en Mhz (izquierdo) como en Khz (derecho), el botón que esta en este sector cambia los valores entre las frecuencias. Los 2 despliegues siguientes pertenecen a los indicadores de navegación con sus respectivos enconders de igual funcionamiento que los anteriores, tal vez se pueda apreciar un botón extra, este sirve para habilitar el funcionamiento de la navegación. El quinto despliegue pertenece al indicador del transpondedor con sus respectivos botones para modificar cada uno de los 4 dígitos. Por último, del lado derecho se encuentra la modulo LCD, el cual muestra la información del piloto automático con sus respectivos botones. En la parte inferior de este se han puesto 6 leds que en la cabina real no existen, pero se utilizan para poder representar el mecanismo de que algunos botones accionan otros. Al final todo este módulo es controlado con la board beagleboneblack.

3.4.2 Modulo de botones.

Este módulo está compuesto por los switches de luces, corriente, tubo pitostatico y la bomba de combustible, tal como se puede apreciar en las siguientes imágenes:

Figura 3.12 módulo de botones (vista frontal).

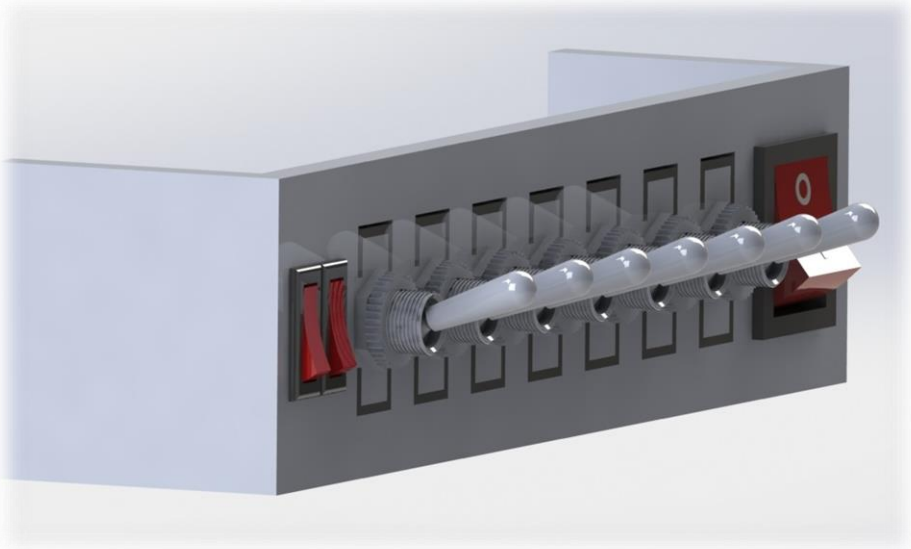

Figura 3.13 módulo de botones (vista lateral).

Del lado izquierdo se puede observar 2 botones rojos que pertenecen a la batería y alternador, el botón del lado derecho pertenece al suministro de corriente en la aviónica y los switches centrales perteneces a las luces. Este módulo es controlado por la board STM32 núcleo, debido a que la beaglebone después del cableado con el módulo de comunicaciones no quedan pines disponibles, los únicos disponibles no se pueden modificar porque pertenecen a la ranura SD y al bus de datos perteneciente a la memoria interna del sistema, ambos son utilizados para el funcionamiento normal de la board, se ha deshabilitado la conexión HDMI para poder disponer de más pines pero ni así se llega a suplir los 12 pines faltantes, la próxima sección se enfocara un poco más las conexiones con los pines utilizados en cada tarjeta. Ya que es necesario el uso de la board núcleo, se decide utilizar esta misma en las conexiones de los potenciómetros lineales perteneciente a la etapa de potencia, no se implementan en la board beaglebone debido a que estos pines deben trabajar como máximo a un voltaje de 1.7 voltios, por ende, se debería diseñar un regulador de voltaje para esto y seria complicar un poco esta versión beta del simulador.

3.4.3 Modulo de potencia.

Este último modulo integra 2 componentes de la etapa de potencia de la aeronave y un switch para el control de los flaps en las alas. Se dispone de 2 palancas con un pequeño desplazamiento en un solo eje para cambiar el porcentaje de combustible o aire que entra en la combustión del motor, esto se hace anclando estas 2 perillas a los potenciómetros lineales para poder obtener bastante rango de datos y así convertir el desplazamiento real por el deseando por parte del simulador. Estos 3 componentes son conectados en la board STM32 núcleo.

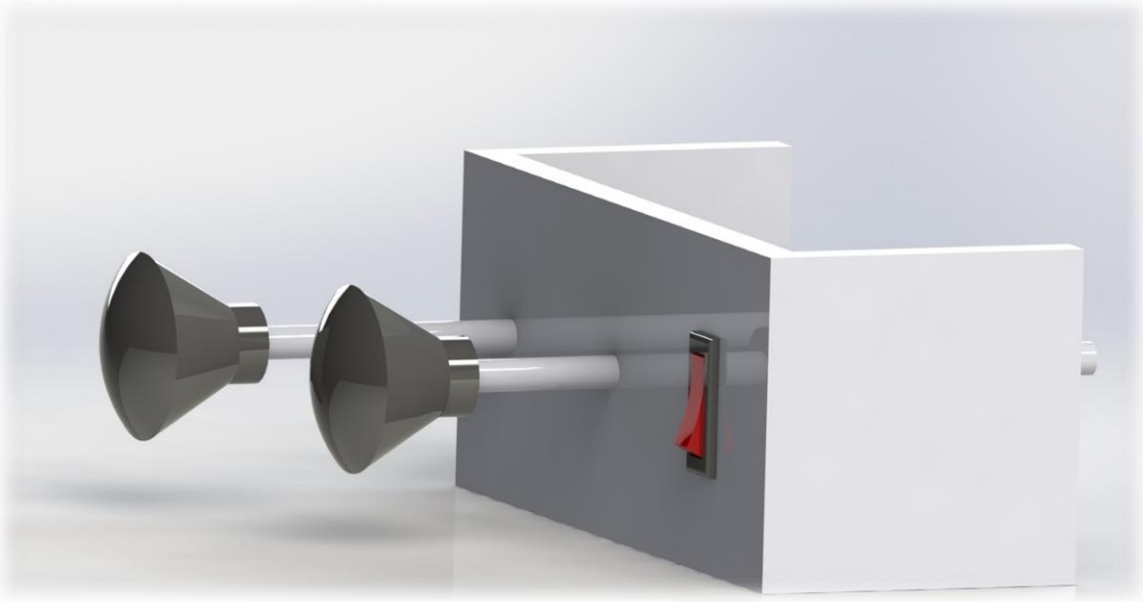

Figura 3.14 módulo de potencia (vista lateral).

3.5 Conexiones entre Pines y Componentes.

En las siguientes 3 tablas se puede observar las conexiones entre cada uno de los componentes electrónicos y los pines pertenecientes tanto a la beaglebone como a la board nucleo. En el caso de la Beaglebone más de 18 pines de 96 pertenecen a voltaje y tierra, 10 más pertenecen al bus datos de la memoria SD, recordando que el sistema operativo corre sobre la SD y no sobre la memoria interna, estos pines no pueden ser modificados. Los 7 pines del ADC no son empleados debido a que solo función bajo un cierto voltaje y de ser mayor este, se dañara el pin. Se debe garantizar que los 8 pines que funcionan con los encoders rotativos, internamente manejen interrupciones para su buen funcionamiento. Con respecto a los pines SPI, hay disponibilidad de 2 puertos, pero se hace uso de uno ya que los módulos permiten la comunicación en cascada por medio de los 3 hilos. Es así como en tan solo la Beaglebone bone se hacen uso de más de 48 pines para el funcionamiento del módulo de comunicaciones.

P9							P8					
	Función	Pines Físicos		Función				Función	Pines Físicos		Función	
	DGND	1	2	DGND				DGND	1	2	DGND	
	VDD 3.3V	3	4	VDD 3.3V				MMC1_DAT6	3	4	MMC1_DAT7	
	VDD 5V	5	6	VDD 5V				MMC1_DAT2	5	6	MMC1_DAT3	
	SYS 5V	7	8	SYS 5V			LED_1	GPIO_66	7	8	GPIO_67	LED_2
	PWR_BUT	9	10	SYS_RESET			LED_3	GPIO_69	9	10	GPIO_68	LED_4
ENC1_CLK	UART4_RXD	11	12	GPIO_60	ENC2_CLK		LED_5	GPIO_45	11	12	GPIO_44	LED_6
ENC1_DT	UART4_TXD	13	14	EHRPWM1A	ENC2_DT		LCD_RS	EHRPWM2B	13	14	GPIO_26	LCD_EN
ENC3_CLK	GPIO_48	15	16	EHRPWM1B	ENC3_DT		LCD_D4	GPIO_47	15	16	GPIO_46	LCD_D5
SPI_CS	SPIO_CSO	17	18	SPIO_D1	SPI_DI		LCD_D6	GPIO_27	17	18	GPIO_65	LCD_D7
	I2C2_SCL	19	20	I2C2_SDA				EHRPWM2A	19	20	MMC1_CMD	
SPI_D0	SPI0_D0	21	22	SPIO_SLCK	SPI_CLK			MMC1_CLK	21	22	MMC1_DAT5	
ENC4_CLK	GPIO_49	23	24	UART1_RXD	TX			MMC1_DAT4	23	24	MMC1_DAT1	
ENC4_DT	GPIO_117	25	26	UART1_RXD	RX			MMC1_DATO	25	26	GPIO_61	
	GPIO_115	27	28	SP11_CSO			BTN_TR_1	LCD_VSYNC	27	28	LCD_PLCK	BTN_PA1
	SP11_D0	29	30	GPIO_112	BTN3		BTN_TR_2	LCD_HSYNC	29	30	LCD_AC_BIAS	BTN_PA2
	SP11_SCLK	31	32	VDD_ADC			BTN_TR_3	LCD_DATA14	31	32	LCD_DATA15	BTN_PA3
	AIN4	33	34	DGND			BTN_TR_4	LCD_DATA13	33	34	LCD_DATA11	BTN_PA4
	AIN6	35	36	AIN5			BTN_TR_5	LCD_DATA12	35	36	LCD_DATA10	BTN_PA5
	AIN2	37	38	AIN2			BTN_TR_6	LCD_DATA8	37	38	LCD_DATA9	BTN_PA6
	AIN0	39	40	AIN1			BTN_TR_7	LCD_DATA6	39	40	LCD_DATA7	BTN_PA7
BTN1	GPIO_20	41	42	ECAPWM0	BTN2		BTN_TR_8	LCD_DATA4	41	42	LCD_DATA5	BTN_PA8
	DGND	43	44	DGND			BTN_TR_9	LCD_DATA2	43	44	LCD_DATA3	
	DGND	45	46	DGND			BTN_TR_10	LCD_DATA0	45	46	LCD_DATA1	

Tabla 3.1: esquema de pines de la board BeagleBone Black para la regleta de pines P9 y P8.

CONVENCIONES	
CÓDIGO	**SIGNIFICADO**
ENCX_CLK, ENCX_DT	Pines de los enconders
SPI_X	Pin SPI de los módulos 7 segmentos
BTNX	Botones de la parte superior del panel de comunicaciones
RX, TX	Pines de transmisión y recepción
LED_X	Pines botones led piloto automático
LCD_X	Pines del módulo LCD del piloto automático
BTN_TR_X	botones del transpondedor
BTN_PA_X	botones piloto automático

Tabla 3.2: convenciones de los periféricos para la board BeagleBone Black

Con respecto a la tarjeta núcleo, al ser un microcontrolador de combate, no hay tantas restricciones por parte de los pines, lo interésate es que la mayoría de los GPIOs se pueden manejar con interrupciones para un mejor control de periféricos. Al necesitar solamente 12 pines y 2 ADC se utiliza solo una regleta de pines de la tarjeta, tal como lo muestra la siguiente tabla:

	función	pines físicos		función	
	PC_10	1	2	PC_11	
	PC_12	3	4	PD_2	
	3,3V	5	6	ESV	
	BOOT0	7	8	GND	
	PF_6	9	10		
	PF_7	11	12	IOREF	
SW_7	PA_13	13	14	NRST	
SW_8	PA_14	15	16	3,3V	
SW_9	PA_15	17	18	5V	
	GND	19	20	GND	
SW_10	PB_7	21	22	GND	
SW_11	PC_13	23	24	VIN	
SW_12	PC_14	25	26		
	PC_15	27	28	PA_0	SW_4
	PF_0	29	30	PA_1	SW_3
	PF_1	31	32	PA_4	SW_2
	VBAT	33	34	PB_0	SW_1
SW_6	PC_2	35	36	PC_11	POT_2
SW_5	PC_3	37	38	PC_0	POT_1

Voltaje, Tierra, Reset
Pines digitales
pines analógicos

CONVENCIONES	
CÓDIGO	**SIGNIFICADO**
POT_X	Potenciómetro lineal para la etapa de potencia
SW_X	switches utilizados para los botones

Tabla 2.3: convenciones de los periféricos para la board STM32 Núcleo.

Tabla 3.4: esquema de pines de la board STM32 Núcleo para la regleta CN7.

3.6 Esquemáticos de los circuitos para el módulo de comunicaciones.

Debido a que el panel de comunicaciones tiene bastantes componentes, se decide realizar un circuito para evitar problemas de ruido con la comunicación SPI y acomodar los componentes electrónicos de pulsadores y leds con sus respectivas resistencias de carga. Se decide hacer uso del software Eagle para la elaboración de cada una de las piezas y pistas que van a ir en el circuito, ya que este se va a realizar de manera artesanal, se decide partir en 2 piezas debido a que comercialmente se consiguen PCB (baquelita) de 10x20 cm, entonces en una se acopa todo lo relacionado a la navegación y comunicaciones, y en la otra parte se acopla el transpondedor y el piloto automático con sus respectivos botones. Se anexa cada una de las piezas elaboradas, las conexiones entre los componentes y las pistas de los circuitos en las siguientes imágenes: ANEXO C.

CAPÍTULO 4: ADQUISICIÓN Y PROCESAMIENTO DE LA INFORMACION

4.1 Configuración del entorno de programación de la tarjeta de desarrollo BeagleBone Black.

La tarjeta BeagleBone Black funciona con un sistema operativo en tiempo real, eso indica que para su funcionamiento se deben hacer ciertas configuraciones e instalación de algunos paquetes para su buen funcionamiento. Cabe aclarar que, al ser un dispositivo con características básicas, se recomienda trabajar con alguna distribución Linux. La distro de Linux que cuenta con más soporte al día de hoy y que le siguen haciendo actualización al kernel del sistema operativo es DEBIAN por ende, se va a trabajar con dicha distribución y los pasos de configuración de la tarjeta se van realizar con dicho sistema.

4.1.1 Instalación del sistema operativo: DEBIAN.

Primeramente, se debe descargar una imagen o instalador del sistema operativo a utilizar, para este caso la versión del sistema operativo que se va instalar es Debian 9.5 2018-10-07 y se puede descargar del siguiente enlace: *https://beagleboard.org/latest-images*. El sistema operativo se instala en una memoria microSD, para ello se utiliza un software llamado *Win32 disk* el cual permite crear instaladores en cualquier tipo de memoria externa. Para acceder a la tarjeta de desarrollo se debe conectar a una red de área local, esta le asigna un IP de funcionamiento y por medio de esta se podrá acceder a la tarjeta, esta dirección IP por lo general tiene una estructura de tipo 192.168.X.XX donde X es un valor por defecto que asigna el router al cual está conectada dependiendo del número de dispositivos. Para acceder al sistema por medio de esta IP se utiliza el software *Putty*, solamente con saber la IP, el usuario y el password se podrá acceder a la consola de comandos, tanto el usuario como el password que viene pre configurados de fabrica es Debian para ambos.

4.1.2 Librerías y paquetes de instalación.

Una vez teniendo el sistema operativo andando se deben utilizar ciertos comandos para actualización e instalación de paquetes para cada uno de los componentes que son utilizados para el funcionamiento del simulador. A continuación, se muestras algunos comandos en el orden que deben ser implementados y para que funcionan:

- apt-get update: este comando actualiza la lista de paquetes disponibles y sus versiones en los servidores con repositorios, no instala o actualiza ningún paquete.
- apt-get upgrade: una vez el comando anterior descarga la lista del software y versión disponible, este actualiza los paquetes instalados en el sistema operativo.
- apt-get install: este comando se utiliza para instalar cualquier paquete por nombre.
- apt-get install python python-pip: cualquier distro de linux por defecto ya trae instalado la versión 2.7 de lenguaje de programación python, en el capítulo 2 se habló un poco de las ventajas y características de trabajar con este lenguaje. Pip es un

sistema que gestiona paquetes utilizado para instalar y administrar software escrito en python. [23]

- pip install spidev: este paquete instala un módulo para interfaz dispositivos SPI a través del driver spidev del kernel de Linux, es capaz es definir, abrir, enviar y modificar velocidad de transmisión del puerto. [24]
- pip install max7219: este comando instala una biblioteca para manejar el driver max7219 que maneja los 8 despliegues de 7 segmentos de cada módulo a través del puerto SPI, esta librería soporta el manejo de hasta 9 despliegues en cascada, manejo de brillo y cualquier tipo de carácter. [25]
- pip install pyserial: esta librería permite el acceso y control de los puertos seriales del hardware. [26]
- pip install adafruit_BBIO: es una librería que permite facilitar un poco el llamado de cada periférico, GPIO, PWM, ADC, I2C, SPI, UART, cada pin tiene una clave dependiendo del propósito para el cual es utilizado y un número para su funcionamiento, el lado derecho de la regleta de pines se conoce como P8 y el lado izquierdo como P9. No todos los pines están disponibles, los pines reservados perteneces al HDMI y al módulo flash eMMC. En el caso del simulador de vuelo, en la BeagleBone solo se utilizan los periféricos GPIO, SPI y UART.

  Para el caso de los GPIO, la librería permite configurar los pines como entradas, salidas, con valores en alto, bajo y permite también configurar si hay eventos de subida o de bajada lo que se conoce como RAISING y FALLING.

  Para el caso del SPI, dicha librería permite configurar cada uno de los pines para la comunicación en este con los despliegues de 7 segmentos.

  En el caso del UART, la librería permite la interacción con los 5 puertos seriales, configurando la velocidad, envió y recepción de datos. Además, puede emular un puerto virtual llamado *minicom* para hacer cualquier tipo de prueba sin necesidad de programarlo. [27]
- pip install char-lcd: esta librería permite configurar los pines y la información que se va desplegar en el caso del simulador en una pantalla de 16x2 pixeles. [28]

Con este conjunto de paquetes y librerías, la tarjeta de desarrollo ya quedara configurada para programar cada componente que lleva el simulador de vuelo.

4.2 Configuración del entorno de programación de la tarjeta de desarrollo STM32F4 Núcleo.

Al haber tantos componentes en un simulador de vuelo, y no la cantidad de pines para suplir la necesidad, se opta por utilizar un microcontrolador que respalde a la tarjeta de desarrollo principal con el fin de suplir los pines faltantes. Una buena alternativa a explorar son los microcontroladores ARM pertenecientes a la familia STM32 que hoy en día son utilizados en la industria en la mayoría de tecnología que depende de un microcontrolador para su funcionamiento. El único problema de estos dispositivos, es que dependen de un IDE de desarrollo de pago para su funcionamiento. Por ende, se opta configurar un IDE (Eclipse) Open Source aprovechando que las librerías para cualquier periférico de esta tarjeta, el fabricante las proporciona de manera gratuita por medio de la página web. [29]

4.2.1 Configuración del IDE de desarrollo.

Debido a que la configuración del IDE es un proceso de muchos pasos a seguir, se dejara este subcapítulo en el anexo A.

4.3 Software de interfaz LINK2FS.

Linkfs es un software libre de licencia GNU programado en C# con el SDK SimConnect reference (propiedad de Microsoft) por una empresa neozelandesa, es considerado un complemento de flight simulator x, está diseñado para agregar funcionalidad a una cabina casera tipo low cost por medio de una placa Arduino. Su funcionamiento es básico, permite la lectura de cualquier componente electrónico que tenga alguna manifestación de cambio en su voltaje, corriente o resistencia y para ello tiene parámetros de entrada y salida. Los parámetros de salida por lo general son los indicadores que el simulador de vuelo tiene dentro de cada aeronave como por ejemplo los indicadores de radios, navegación, transpondedor y piloto automático, mientras que los parámetros de entrada son todos los relacionados con botones o actuadores que necesitan de algún cambio real para su activación. Además de estas ventajas, el software también entrega información de ubicación espacial bajo un punto de referencia de la aeronave para poder visualizar los cambios en cualquier de los 3 ejes de traslación y 3 ejes de rotación con el fin de materializar en una estructura de 6 grados de libertad, pero para el caso de esta investigación no se va a tener en cuenta.
El software permite además la conexión por serial de máximo 5 dispositivos, la velocidad a la cual opera él sistema es 115200, se puede configurar el puerto serial de cada tarjeta, tiempo de cada ciclo de operación y tiempo de recarga de cada ciclo. El software se puede descargar por medio del siguiente link: http://www.jimspage.co.nz/link2fs_experts.htm

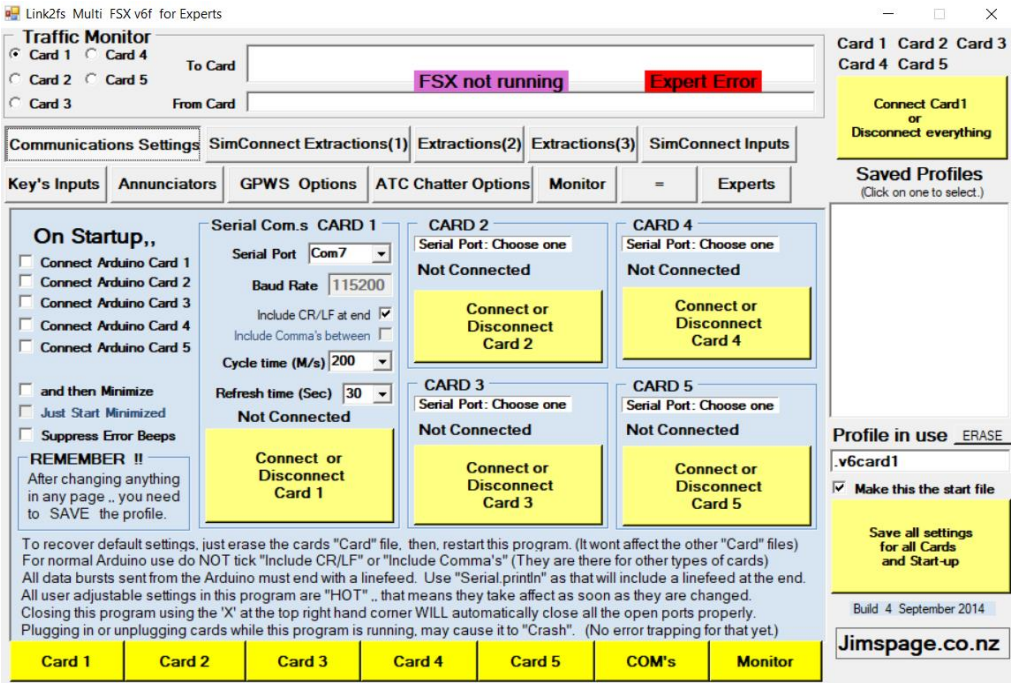

Figura 4.10 interfaz de software Link2fs.

4.3.1 Configuración de parámetros de salida de links2fs para una aeronave Cessna 172.

Los indicadores que se tienen en la cabina de una Cessna 172 son los siguiente, que, además, están representados en la siguiente tabla con cada clave que el software link2fs proporciona:

Indicador tipo salida de Link2fs	Clave	Aparato electrónico en el cual es visualizado
Frecuencia de la comunicación 1	=A	Despliegue de 7 segmentos
Sub-Frecuencia de la comunicación 1	=B	Despliegue de 7 segmentos
Frecuencia de la navegación 1	=E	Despliegue de 7 segmentos
Sub-Frecuencia de la navegación 1	=F	Despliegue de 7 segmentos
Código del transponder	=J	Despliegue de 7 segmentos
Activador del Piloto automático	=a	Pantalla lcd 16x2
Indicador de altitud del piloto automático	=b	Pantalla lcd 16x2
Indicador de velocidad vertical del piloto automático	=c	Pantalla lcd 16x2
Activador HDS del Piloto automático	=i	Pantalla lcd 16x2
Activador de ALT del piloto automático	=k	Pantalla lcd 16x2
Activador APR del piloto automático	=m	Pantalla lcd 16x2
Activador REV del piloto automático	=n	Pantalla lcd 16x2
Activador NAV del piloto automático	=o	Pantalla lcd 16x2

Tabla 4.1: Indicadores para la interfaz de recepción y envió de datos

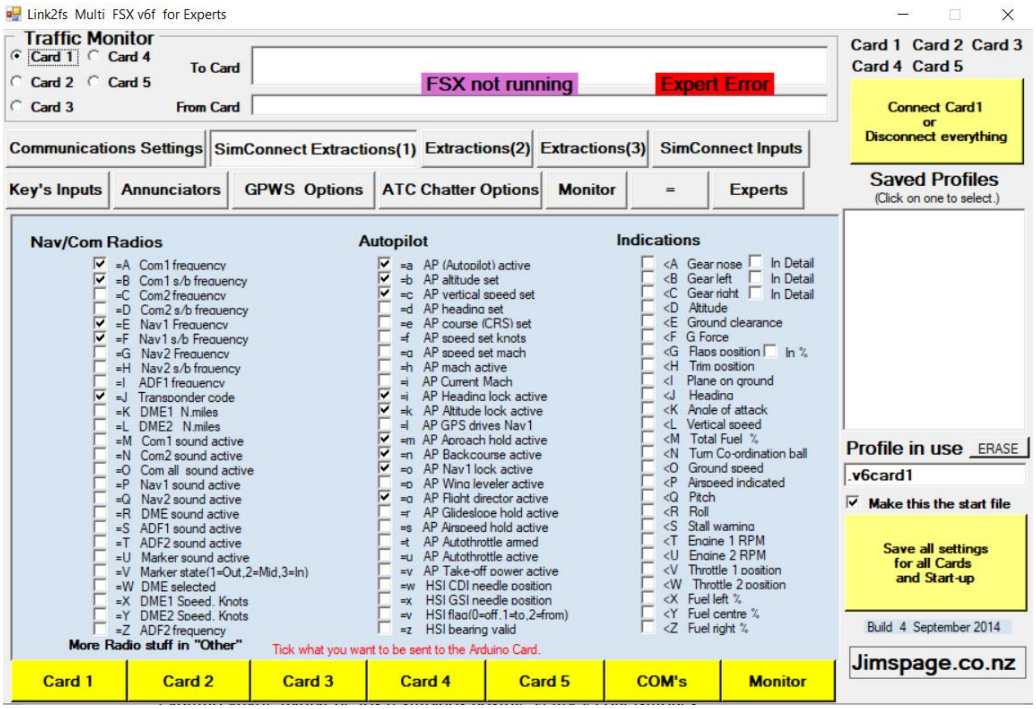

Figura 4.11 indicadores para recepción de datos.

4.3.2 Configuración de parámetros de entrada de links2fs para una aeronave Cessna 172.

Los parámetros de entradas son todos aquellos que tienen un cambio de intervención humana en la cabina de la aeronave Cessna 172, están detallados en la siguiente tabla, donde se muestra: la acción de entrada, la clave entregada por link2fs (cabe aclarar que la clave para los dispositivos pueden ser de tipo: On/Off o valor) y el componente electrónico que genera el cambio de estado, estos cambios se envían por medio del protocolo serial entre la tarjeta de desarrollo y el software link2fs con el simulado flight simulator x en marcha:

Indicador tipo entrada de Link2fs	Clave	Aparato electrónico que genera el cambio de estado
Incrementa/disminuye la frecuencia de comunicación en Mhz	A01/A02	Encoder Rotativo
Incrementa/disminuye la frecuencia de comunicación en 25 Khz	A03/A04	Encoder Rotativo
Incrementa/disminuye la frecuencia de navegación en Mhz	A14/A13	Encoder Rotativo
Incrementa/disminuye la frecuencia de navegación en 25 Khz	A15/A16	Encoder Rotativo
Cambio de frecuencia de comunicación con la standby	A06	pulsador
Cambio de frecuencia de navegación con la standby	A18	pulsador
Activar navegación 1	A48	pulsador
Incrementa/disminuye primer digito del transpondedor	A34/ A35	pulsador
Incrementa/disminuye segundo digito del transpondedor	A36/ A37	Pulsador
Incrementa/disminuye tercer digito del transpondedor	A38/ A39	pulsador
Incrementa/disminuye cuarto digito del transpondedor	A40/ A41	pulsador
Asigna código de emergencia preconfigurado	A42xxxx	pulsador
Enciende/apaga piloto automático	B02/B03	pulsador
Activa HDS del piloto automático	B04	pulsador
Activa ALT del piloto automático	B05	pulsador
Activa APR del piloto automático	B07	Pulsador
Activa REV del piloto automático	B09	pulsador
Activa NAV del piloto automático	B10	pulsador
Incrementa/disminuye valor de la altura en el piloto automático	B11/ B12	pulsador
On/off Swittch aviónica	A431/A430	Toggle swhitch
On/off Swittch alternador	E19	Toggle swhitch

On/off Swittch Batería	E01	Toggle swhitch
On/off Swittch bomba de gasolina	F021/F020	Toggle swhitch
On/off Swittch luces beacon	C421/C420	Toggle swhitch
On/off Swittch luces landing	C431/C430	Toggle swhitch
On/off Swittch luces taxi	C441/C440	Toggle swhitch
On/off Swittch luces navegación	C411/C410	Toggle swhitch
On/off Swittch luces strober	C451/C450	Toggle swhitch
On/off Swittch tubo pitot	C05/C06	Toggle swhitch
On/off Swittch Flaps	C15/14	Rocket Swhitch
Palanca de potencia	C56xxx	Potenciómetro lineal
Palanca de mezcla	C58xxx	Potenciómetro lineal

Tabla 4.2: Código de los indicadores para la interfaz de envío de datos.

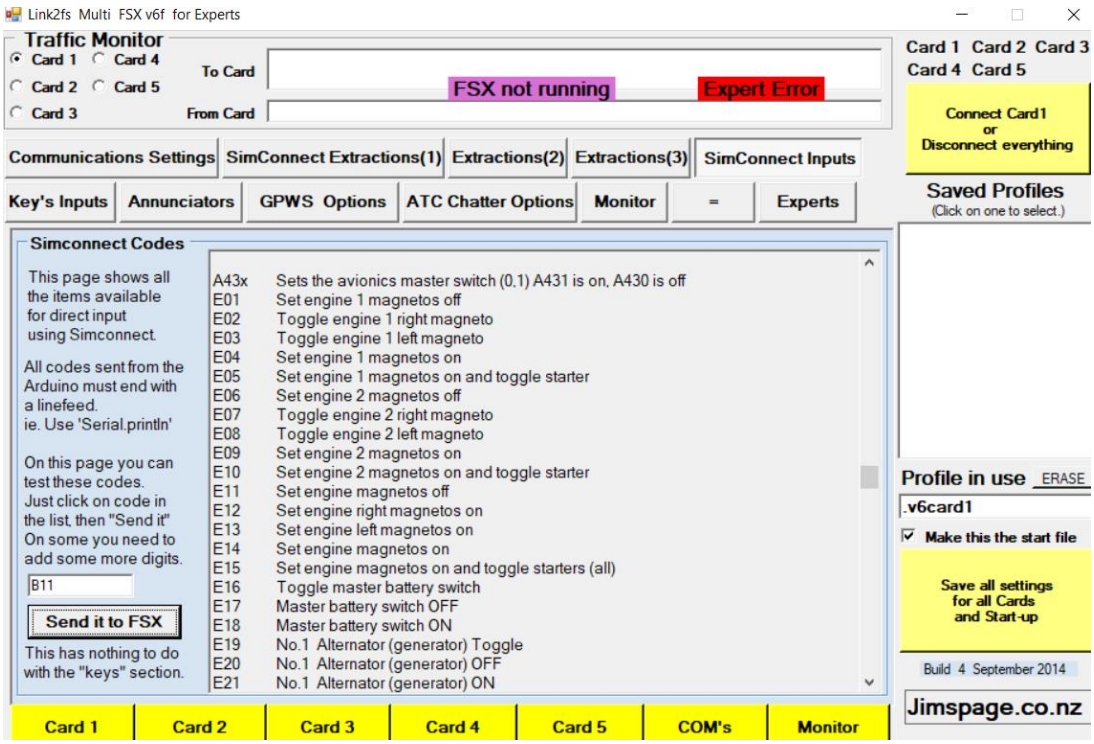

Figura 4.12 indicadores para envío de datos.

4.4 Software para interfaz de indicadores: Air Manager.

Air manager es una aplicación desarrollada por Sim Innovatios que permite crear paneles de instrumentos a la medida para simuladores de vuelo 2D. Se puede ejecutar en la misma computadora donde esté trabajando el flight simulator X o en computadoras separadas por medio de la conexión ethernet, la aplicación reconoce automáticamente el simulador y arrancara de inmediato a funcionar. Air manager, es una aplicación propietaria, lo que indica que se debe hacer un pago para poder tenerla, para este caso de la investigación se habló con

la compañía para poder trabajar con el software de manera gratuita ya que es con fines académicos. La idea de utilizar air manager es hacerle ingeniería inversa para poder crear uno mismo los instrumentos para la cabina del simulador, ya que el software es programado en el lenguaje de programación Lua; este se considera un lenguaje de programación compatible para operar en cualquier plataforma, en el, las variables no tienen tipo, solo datos que pueden ser lógicos, enteros, flotante, cadenas, vectores y conjuntos son apilados en una tabla para su operabilidad. Entonces, si ya se piensa comercializar este simulador de vuelo habría que implementar los visualizadores de los instrumentos utilizando lua y los códigos que air manager proporciona de manera gratuita de cada uno de estos. Los códigos están escritos en lua y existe una gran comunidad que dan soporte para estos, solamente se deben tener las imágenes en PNG de texto, botones, interruptores, etc. y los scripts le darán movimiento a cada componente. Por ejemplo, con este código se muestra lo simple que puede ser darle movimiento de giro a un indicador:

```
-- Añadir Imagen, con nombre y dimensión
Brujula1 = img_add("brujulanav1.png", 10, 10, 100, 200)

-- Rotar brujula1 a 90 grados
rotate(brujula1, 90)

-- Mueve el indicador brujula1 en 10 pixeles en X y Y, deja el ancho y el alto como estaba.
move(brujula1, 20, 20, nil, nil)
```

La aplicación trae en su catálogo una biblioteca bastante amplia de instrumentos de aeronaves de entrenamiento o de aviación comercial, para el caso del Cessna 172 están todos los instrumentos mencionados en el capitulo1 que son los implementados para este simulador. En caso de que cualquier instrumento no esté en las unidades pertinentes o que le falte alguna pieza, este se puede editar y modificar directamente desde air manager.

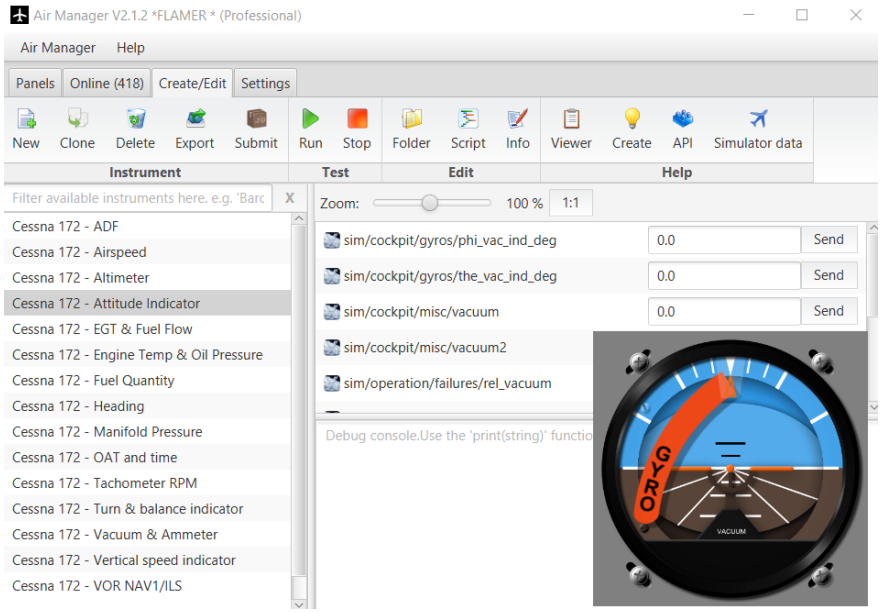

Figura 4.13 interfaz del software air manager.

4.5 Scripts de programación para cada componente electrónico.

En esta sección del capítulo 3 se mostrará un pequeño resumen de la implementación de cada uno de los códigos tanto del filtro que se diseñó para la lectura de cada parámetro de salida de link2fs como los códigos de los parámetros de entrada relacionados con los componentes electrónicos utilizados en la construcción de este simulador, todos estos códigos se cuentan en el anexo B de este documento.

CAPÍTULO 5: RESULTADOS.

5.1 Resultados en la construcción de cada módulo.

En el capítulo 2 se diseñó el modelo de cómo iba organizado cada componente en los módulos para el simulador, en este caso, siguiendo ese concepto e idea se han construido cada uno de los 3 módulos que se habían planeado: módulo de potencia, módulo de botones y módulo de comunicaciones.

5.1.1 Modulo de Potencia.

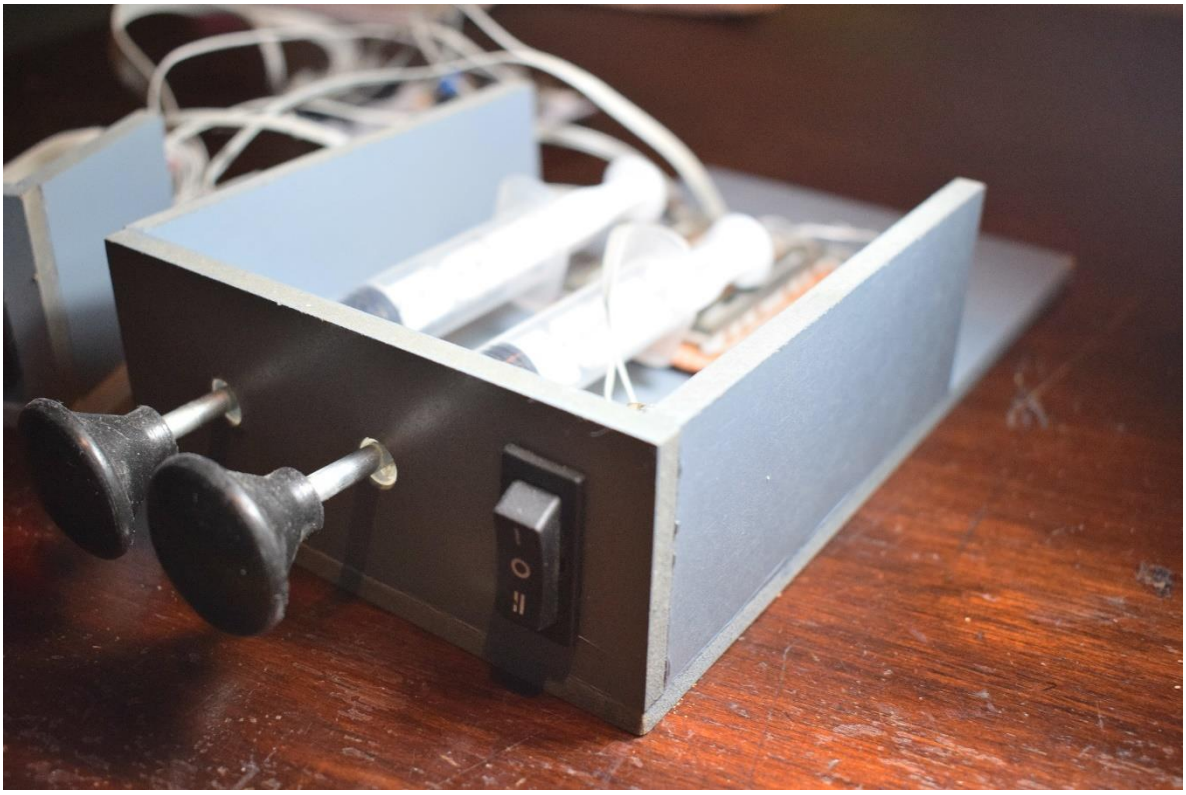

Figura 5.1 resultado del módulo de potencia.

Con respecto al módulo de potencia, para este primer demo, se ha hecho una caja en madera con el fin de poder acoplar los 3 componentes (potencia, mezcla y flaps) conectados a la board de desarrollo stm32f4. La alimentación que requiere este módulo son 5 voltios que se obtienen de la board de desarrollo ya que no hay un alto consumo de corriente y no es necesario una alimentación externa. La etapa de potencia de una aeronave Cessna 172, se debe aplicar una determinada fuerza para mover las guayas que internamente permiten el paso de combustible y aire en el motor, la forma de recrear esa pequeña fuerza que se le debe hacer a estas 2 palancas, es acoplando a cada una de ellas una jeringa de uso médico, ya que el embolo con la goma genera una fricción ideal para representar dicha fuerza, y a su vez esta sea la que transmita el movimiento al potenciómetro lineal, tal como se puede observar en la figura 4.1.

Figura 5.2 resultado del módulo de potencia

El tercer componente que está en la etapa de potencia(flaps), está representada por un switch de 2 pociones de tipo abierto, ya que el módulo real es un poco complejo poder construirlo, este puede bajar o subir los flaps desde una posición central.

5.1.2 Modulo de botones.

Figura 5.3 resultado del módulo de botones.

Con respecto al módulo de botones, se ha construido en una caja de madera y en ella se ha acopado los diferentes tipos de switches similares a los de una aeronave real, la diferencia de unos y de otros en la cabina real, es la cantidad de corriente que deben soportar para su funcionamiento. La idea de que ninguna de las cajas elaboradas tenga una solapa superior es que si no hay mucho espacio para ubicar estos módulos, puedan ir unos encima de otros.

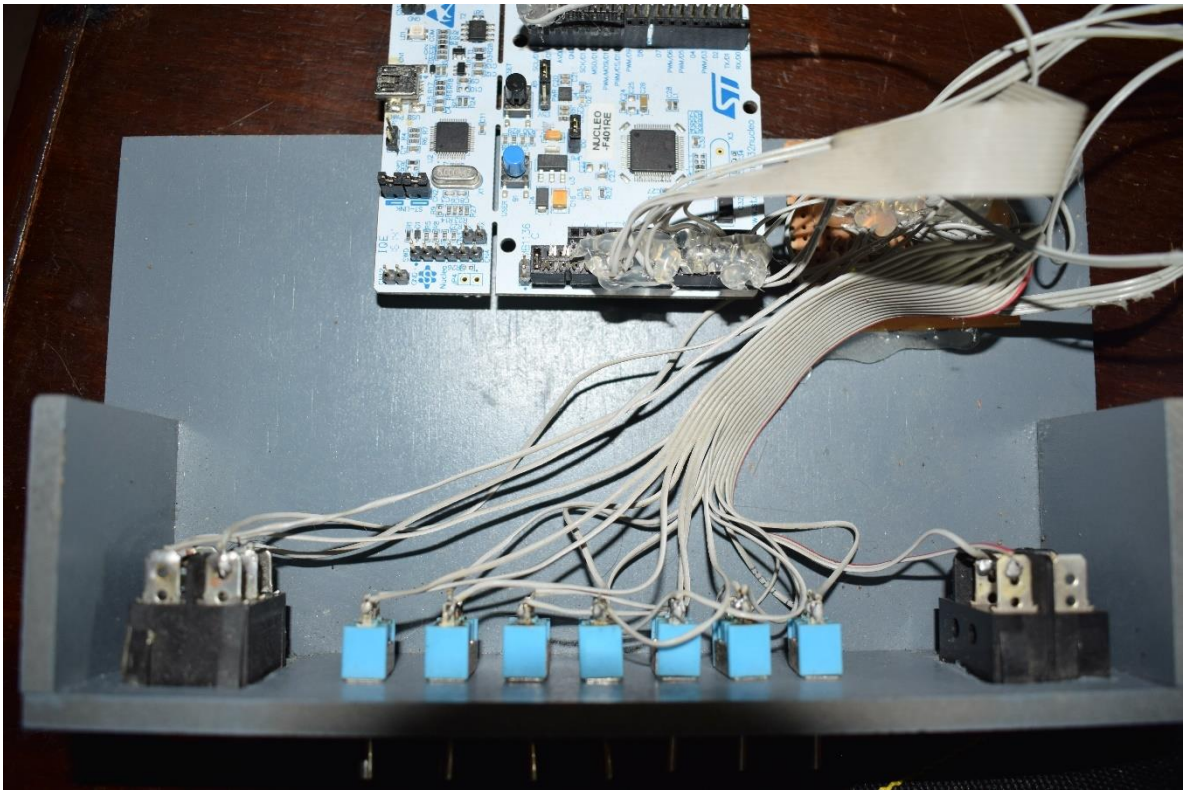

Figura 5.4 conexiones entre la board STM32 y el módulo de botones.

Cada switch va conectado a una resistencia de carga en una pcb y de ahí sale a cada pin de la board stm32f4. El software de adquisición de datos recibe hasta un máximo de 5 dispositivos seriales y la board se conecta a través del FTDI serial al software, dejando 4 dispositivos libres.

5.1.3 Modulo de comunicaciones.

El módulo de comunicaciones, tal vez sea el módulo con mayor trabajo y construcción, este tuvo varios pasos para su elaboración que iba desde diseñar un circuito (esquemáticos completos en el capítulo 2) para acoplar cada uno de los componentes involucrados en este, con el fin de poder eliminar toda clase de ruido, hasta la construcción e implantación de todos los componentes involucrados.

Figura 5.5 primera etapa del circuito del módulo de comunicaciones.

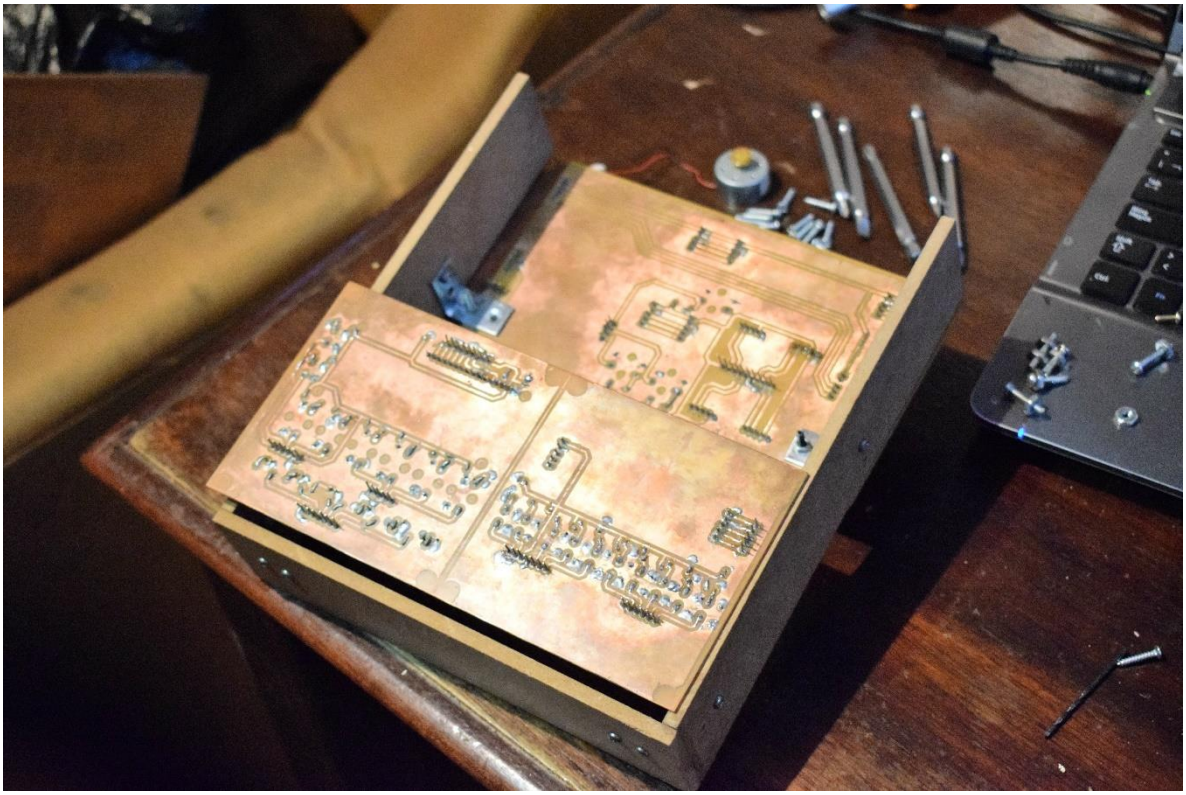

Figura 5.6 pistas del circuito del módulo de comunicaciones.

Al tener una idea y concepto de como deberían ir todos los componentes del módulo de comunicaciones, se construye una caja con las dimensiones de 20x20 Cm (frontal), con una estructura interna que pueda sostener las bases de adentro con el circuito que va al frente de la caja tal como lo muestra a figura 4.6. En la figura 4.1, se puede observar que la parte frontal se ha divido en 2 circuitos, esto es debido a que en el momento de elaborar un circuito de mayor tamaño de manera artesanal, muchas de las pistas no quedaban pegadas en la PCB debido al tamaño. Por esa razón se trabajó con 2 placas de menor tamaño para que cada pista de cobre quedara fija en la PCB.

Después de ciertos arreglos, en los esquemáticos, pistas, soldadura, y componentes el resultado final es el siguiente:

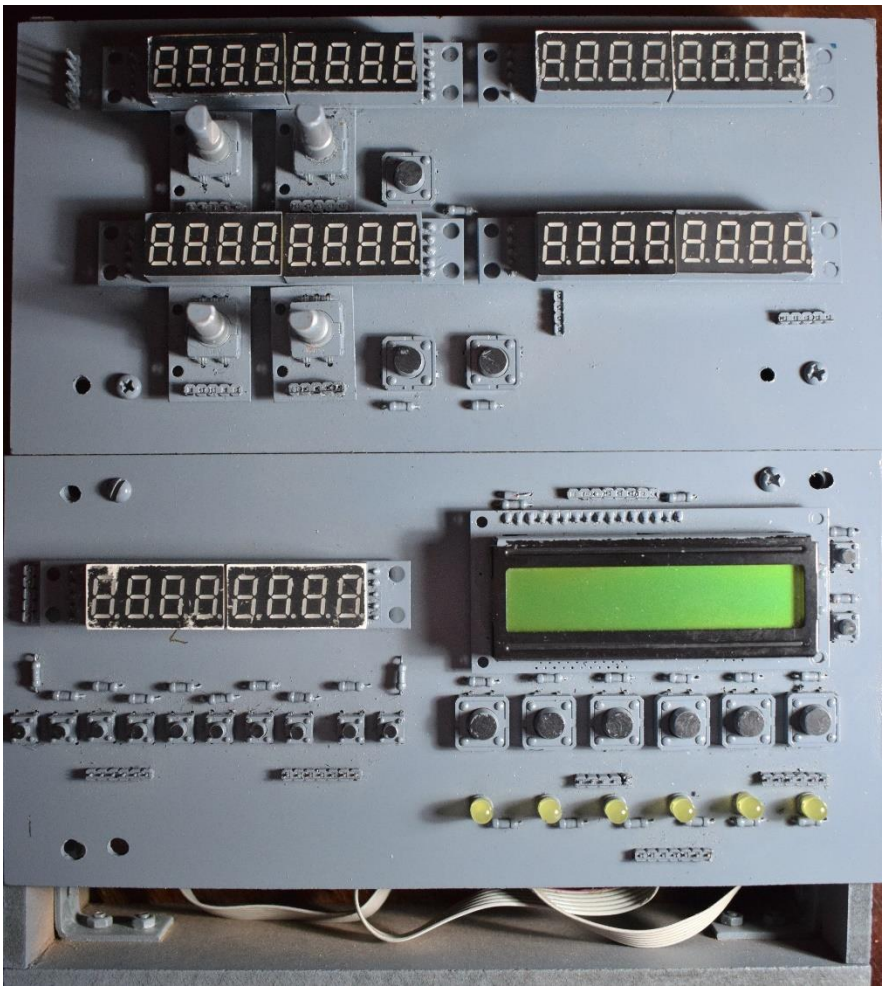

Figura 5.7 módulo de comunicaciones finalizado.

Por el lado frontal como se observa en la figura 4.7, el color gris es debido a que la mayoría de las cabinas en aeronaves de entrenamiento son de este tipo de colores para que la información en los displays sea más llamativa. Por el lado de atrás, se observa en la figura 64, todas las conexiones que se realizaron que a su vez van directamente conectadas a la board de desarrollo BeagleBone Black, y, esta está conectada a un FTDI para hacer la conexión serial con el software de recepción de datos.

Figura 5.8 cableado del módulo de comunicaciones y la board de desarrollo beaglebone black.

En la parte superior de la figura 4.8 se pueden observar 2 cables (naranja y blanco), los cuales son para la alimentación de todos los componentes electicos presentes en este módulo. Debe ser como mínimo de 5 voltios a 2 amperios para un buen funcionamiento de la tarjeta.

5.2 Conexión Final.

Al final las conexiones que quedan para ambas tarjetas con respecto a los módulos, software de aviación y de captura de datos e interfaz gráfica de los instrumentos, serían las siguientes:

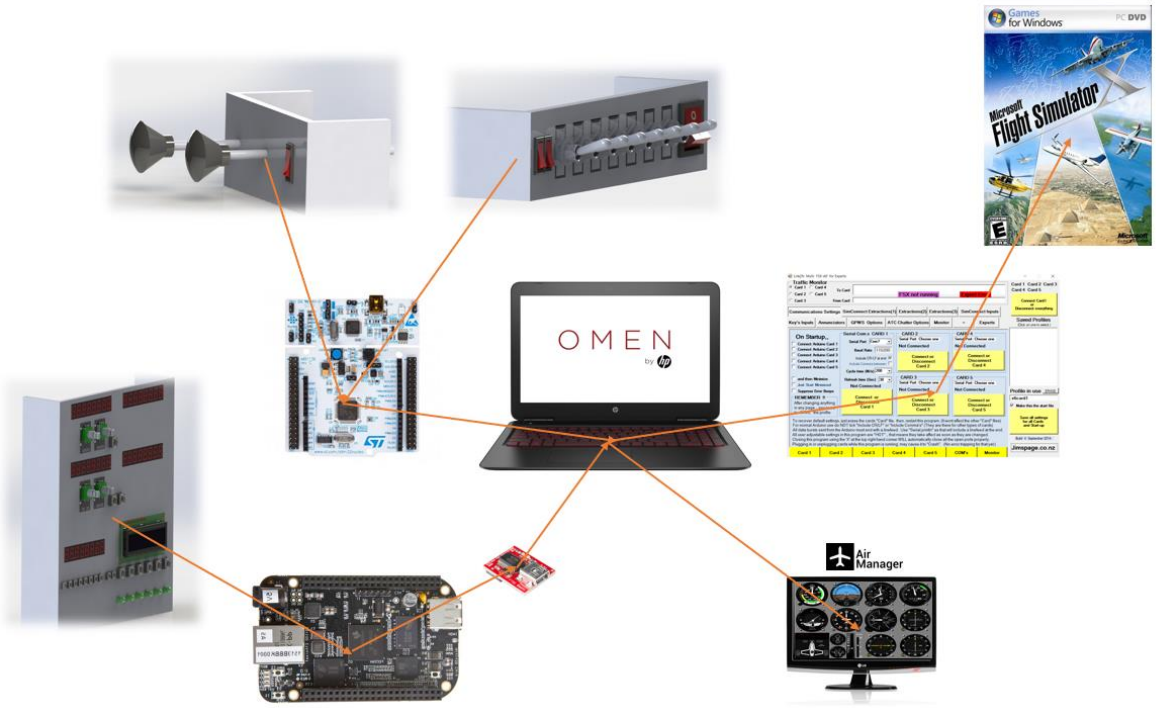

Figura 5.9 esquema final de las conexiones del simulador.

Ambas tarjetas reciben y envían (para algunos componentes) datos que son extraídos o modificados directamente del simulador de vuelo por el medio del programa link2fs, deben tener algún FTDI (internamente o externo) para poder establecer la comunicación. Por otro lado, el software encargado de los indicadores funciona solamente con que el simulador de vuelo este encendido.

5.3 Pruebas de los módulos construidos

Para probar la interfaz del simulador de vuelo realizada en este trabajo de grado, se le ha pedido colaboración a 10 estudiantes de la escuela militar de aviación Marco Fidel Suarez (EMAVI), en un actividad muy básica de vuelo que dure alrededor de 45 minutos, vayan de un punto A (base aérea Cali) a un punto B (zona de entrenamiento permitida en Jamundí) y después de hacer algunos virajes regresa al punto inicial A, sintonizando varias frecuencias de comunicación y que en medio del vuelo se presente alguna anomalía que sea representada por medio del transponer, por último, se hará una aproximación a tierra utilizando el piloto automático y el aterrizaje se hará de manera manual. Se llevarán a cabo cada uno de los checklist (o lista de comprobación, son cartas que tienen instrucciones que dependiendo de punto del vuelo se deben cumplir, como por ejemplo las luces, los flaps, etc.) obligatorios. La torre de comunicación será simulada por un instructor de vuelo de la escuela. Por cuestiones de seguridad y protocolos no se podrán publicar ni fotografía de los estudiantes ni nombres.
La idea es analizar cuanto tiempo gastan en realizar el vuelo descrito anteriormente con 4 dispositivos diferentes: mouse, teclado y mouse, interfaz hecha para este simulador y por el

ultimo en el simulador que dispone la fuerza área de esta aeronave. Al final se hará una pequeña encuesta que cada estúdiate llenará para hacer un análisis de datos.
Debido a que es un base militar y el espacio por el cual sobrevuelan en sus prácticas es confidencial, no están habilitadas las cartas aeronáuticas para personas civiles, se ha diseñado un pequeño plan de vuelo para poder llevar a cabo esta práctica tal como se muestra en la figura 4.10, 4.11 y 4.12:

Figura 5.10: Salida de la base EMAVI.

Figura 5.11: viraje en zona de entrenamiento.

Figura 5.12: ruta establecida para el vuelo de prueba.

Se completan los datos del tiempo en cada una de las interfaces evaluadas por cada uno de los 10 estudiante en la siguiente tabla:

	TIEMPO (mins) DE VUELO EN LA PRACTICA (Promedio 45 minutos)			
Piloto	Mouse	Teclado y mouse	Simulador tesis	Simulador EMAVI
1	64.3	62.3	49.3	47.4
2	67.5	65.2	47.6	45.7
3	60.7	59.5	46.1	45.2
4	60.8	58	46.4	45.1
5	69	63.7	48.1	46.5
6	71.2	62.4	48.8	47
7	73.5	65.1	51.3	49.8
8	65.5	60.3	50	47.1
9	66.1	61.2	47.1	45.5
10	62.4	57.7	45.5	43.2

Tabla 5.1: datos del tiempo de cada vuelo realizado por los pilotos.

Para entender de manera más fácil estos datos consignados, se utiliza la siguiente grafica que en el eje X están los estudiantes y en el eje Y esta el tiempo que cada uno gastó en las 4 interfaces (4 líneas de diferentes colores) propuestas para la comparativa de cada una.

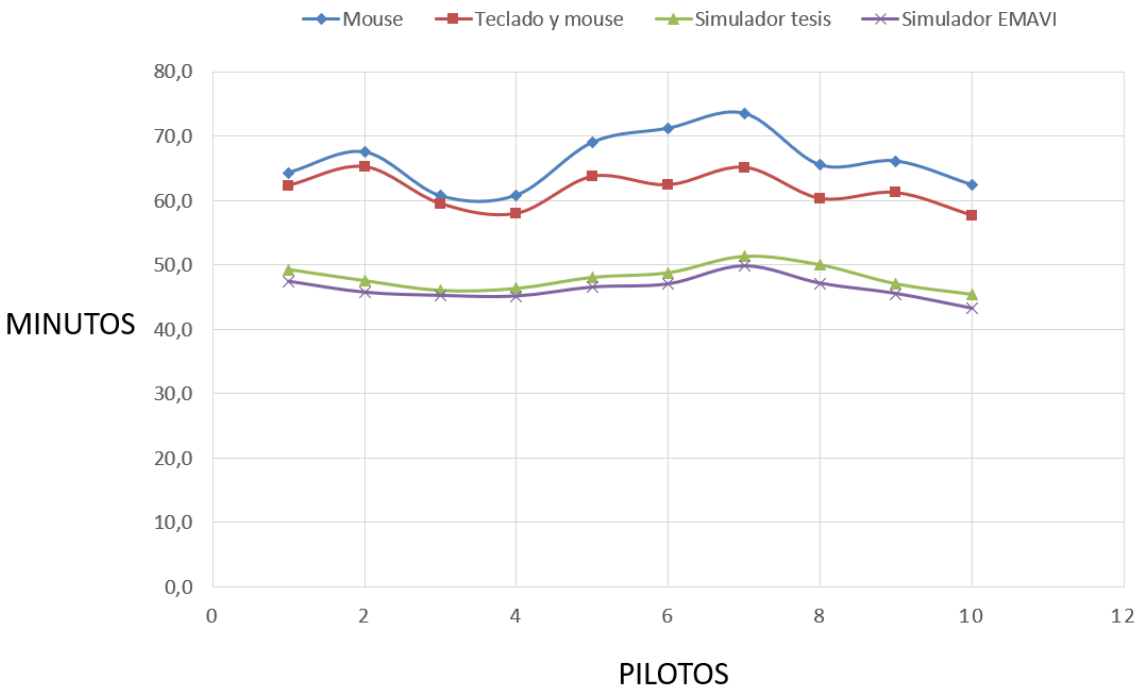

Figura 5.13: resultados del tiempo de los diferentes vuelos realizados por los pilotos.

Con respecto a la gráfica x se puede observar que las herramientas básicas que un computador tiene, como es el mouse y teclado para este tipo de software no es lo suficientemente efectivas en el entrenamiento de un piloto ya que no hay similitud de estas interfaces con la de una cabina de una aeronave, además que debe haber una pequeña curva de aprendizaje de comando y números y tener la destreza de hacer cambio de manera muy rápida en los actuadores y botones sin perder el rumbo de la aeronave.
Por otro lado, la interfaz hecha en este trabajo de investigación entrega casi el mismo desempeño que el simulador que tienen en la base aérea, tal vez lo que hace la diferencia entre ambas interfaces es la distribución de controles y botones, en varios paneles como esta este trabajo con respecto al aspecto de una cabina real que tiene el simulador de EMAVI.
Después de haber realizado cada una de las 4 practicas, se les entrego una cuesta con 5 preguntas a cada uno de los estudiantes, con el fin de hacer un análisis cuantitativo de que tan viable seria implementar este sistema de bajo costo en los diferentes sitios de entrenamiento.

Leyenda:
1 = Totalmente en desacuerdo o valoración muy baja.
2 = Desacuerdo o valoración baja.

3 = Neutral.
4 = De acuerdo o valoración alta.
5 = Totalmente de acuerdo o valoración muy alta.

	1	2	3	4	5
El simulador del trabajo de la investigación funciona de igual forma que el simulador de la base EMAVI (sin lags ni bloqueos).					
La distribución de los botones, actuadores e instrumentos es la adecuada.					
Es necesario un tiempo de adecuación para el uso del simulador del trabajo de grado.					
¿Compraría este simulador por menos de 800 dólares?					
Valora tu grado de satisfacción con el simulador del trabajo de grado.					

Tabla 4.2: encuesta realizada a los pilotos después del vuelo.

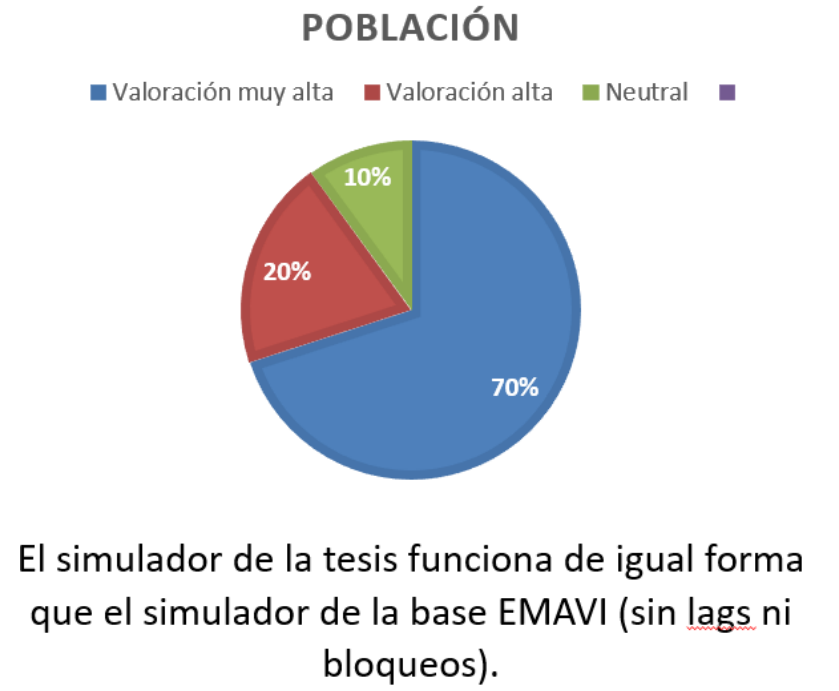

Figura 5.14: porcentaje de población en la funcionalidad del simulador.

En la figura X se puede observar que el 80% de estudiantes afirman que el simulador construido para este trabajo de grado funciona de igual forma que el simulador profesional

de la base, el 20% réstate de la población afirma haber tenido que interactuar 2 a 3 veces en algunos botones para su funcionamiento, debido a que las pistas y soldadura de algunas piezas están en mal estado debido a la oxidación del cobre ya que la forma en que se fabricó la PCB es de manera artesanal.

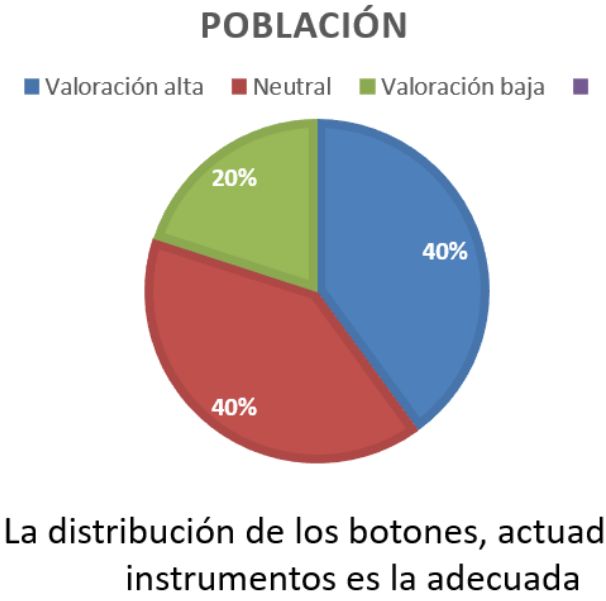

Figura 5.15: porcentaje de población en la distribución de los botones.

Debido a que la distribución de actuadores y botones no es la misma de una cabina real de una aeronave Cessna 172, las opiniones con respecto a esta pregunta (figura x) están un poco más polarizadas, los pilotos con mayor experiencia que equivale al 40% han considerado que el simulador está bien, pero el otro 40% y un 10%, no comparten mucho la distribución en paneles de los componentes.

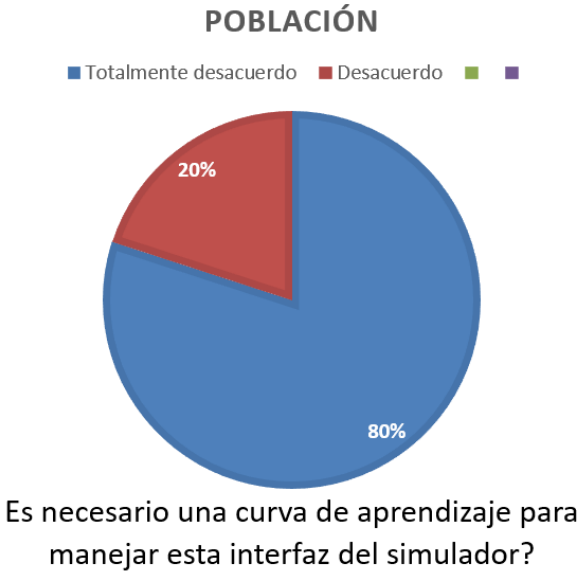

Figura 5.16: porcentaje de población en tiempo de entrenamiento necesario para el simulador.

Por la misma experiencia de los pilotos, ya se han adecuado a las cabinas de las aeronaves, por ende, esta pregunta va un poco ligada a la anterior, debido a la distribución de los actuadores y botones muchas acciones los pilotos las hacen de manera automática, debido a que han utilizado un sistema nuevo deben hacer un pequeño análisis y estudio del simulador, mientras que los pilotos más experimentados afrontan con mayor facilidad los cambios.

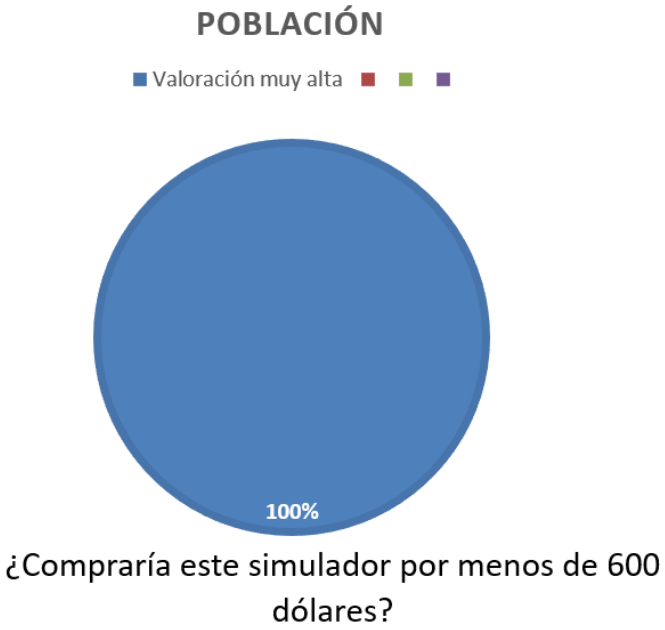

Figura 5.16: porcentaje de población del interés de compra del simulador.

Al comentar que el diseño y la construcción de esta interfaz de simulador ronda los 800 dólares, todos los pilotos estuvieron de acuerdo en una compra inmediata debido a que podrían realizar prácticas en cualquier momento, debido que los simuladores de la base (son 2) en la semana está en completo funcionamiento por los diferentes pilotos que tiene la institución. La única recomendación que sugieren es hacer la distribución de botones y actuadores lo más parecido a la cabina de la aeronave.

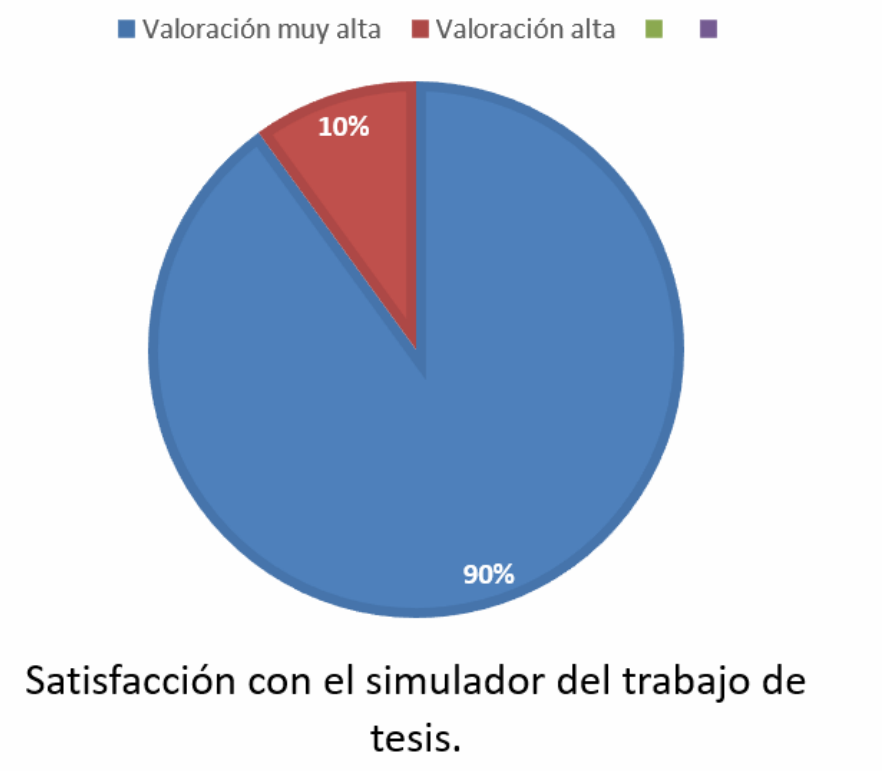

Figura 5.17: porcentaje de población en satisfacción del simulador.

Pese a la distribución que tienen los paneles diseñados y construidos para este trabajo, en la población estudiada, los estudiantes demuestran una gran satisfacción de uso con en el funcionamiento de esta interfaz del trabajo de grado.

Por otro lado, lo módulos también fueron evaluados por un experto en el tema (piloto comercial y director de la tesis, Mario Andrés Córdoba) en vuelos cortos, el cual colocaba en funcionamiento cada uno de los actuadores e indicadores presentes en los módulos, dando una excelente respuesta en relación a la acción y reacción de cada componente de manera inmediata, tanto en los dispositivos como en los indicadores de instrumentos. Por parte del código de programación, funciona correctamente para cualquiera de las 2 boards de desarrollo, tal vez de parte de la BeagleBone, se esperaría algún bug o lag ya que además de correr el archivo tiene un sistema operativo funcionando en paralelo y este en algunos casos puede dar prioridad a otros procesos; se carga un archivo que esta siempre a la escucha de todos los datos recibidos y enviados sin colgarse en ningún momento en alguna de las pruebas.

Tal vez el único problema presente en esta etapa del simulador es que cualquiera de los 3 módulos debe estar al menos sujeto o atado en alguna superficie para poder accionar cualquier actuador o sensor sin presentar alguna incomodidad. Por otro lado, al utilizar en gran parte módulos comerciales, algunos tienen componentes adicionales (diodos, transistores, reguladores, pines de más, etc.) que no son utilizados o que en algunos casos pueden generar mayor consumo en la potencia o perdida de información.

Por parte del módulo de potencia, el sistema de fuerza artesanal funciona bástate bien, los datos esperados tanto en la mínima posición de la perilla representa el 0% tanto del combustible como del paso del aire, como en la posición máxima de esta que representa el 100% de potencia o de aire. El sistema de flaps funciona bastante bien, el único inconveniente es que no hay un visualizador real para saber el ángulo en el que se encuentran, por otro lado, este problema se puede solucionar colocando un visualizador digital en la pantalla de indicadores.

Por parte del módulo de botones, es el más básico de todos, funcionan sin problema a la velocidad que se cambien el estado de cada uno de estos. El único problema respecto al funcionamiento de una cabina real, es que el botón de alternador funciona bajo un componente mecánico, si la batería no está activada este botón no debería funcionar. Al disponer de este funcionamiento mecánico, se debe arreglar dicho problema por medio de una lógica aplicada en el código.

Por parte del módulo de comunicaciones, los problemas presentes son de tipo físicos que de funcionamiento. Por ejemplo, al utilizar los módulos de encoders rotativos, algunos se dañaron por exceso de uso, desprendiéndose de la pcb que venían de fábrica. Con respecto al circuito elaborado, en algunos casos por demasiado movimiento en los componentes algunas pistas se levantan, lo que generaban daños que se reparan con cables superficiales tal como lo muestra la figura 4.18, generando ruido, en algunos casos se pueden apreciar en los despliegues 7 segmentos donde se encendían todos los números. Con respecto al piloto automático, el sistema que reemplaza a los botones físicos funciona muy bien representados por los leds que hay debajo de cada botón.

Figura 5.18: daños en la PCB

De esta manera se tiene un primer demo de una parte de la cabina de una aeronave Cessna 172 que puede ser utiliza en una escuela de aviación supliendo las necesidades más básicas para un piloto con respecto al desplazamiento y las comunicaciones de la aeronave.

4.4 Costos del demo de la aeronave Cessna 172.

Con respecto a los costos que tiene este demo solo se tendrán en cuenta los materiales y las horas de construcción en la siguiente tabla, se desprecia el valor cobrado por parte de ingeniería aplicada en el desarrollo del software y del código, ya que este solo se elabora una vez y con respecto a los problemas que vayan apareciendo, se irán modificando.

Ítem	Cantidad	Costo por unidad	Total (más envíos)
Board BeagleBone Black	1	175.000	185.000
Board Stm32F4 Núcleo	1	99.300	109.300
FTDI Sparkfun	1	97.990	107.990
Despliegues 7 segmentos, 8 despliegues	5	12.000	70.000
Encoders Rotativos	4	6.000	34.000
LCD 16x2	1	12.000	12.000
Potenciómetros Lineales	2	12.000	34.000
Rocket Switch	1	2.400	2.400
Toggle Switch	7	500	3.500
Action Switch	2	4.000	8.000
Pulsadores	20	400	8.000
Leds	6	300	1.800
Resistencias	32	50	1.600
Correa de cables	2	4.000	8.000
Jumpers	3 regletas de 40 pines	1.50-0	4.500
PCB perforada 5x5 cms	2	2.000	4.000
PCB sin perforar 10x20 cms	2	4.500	9.000
Jeringas	2	400	800
Soportes Estructura	6	9.000	19.000
Madera Estructuras	3 módulos	2.000	8.000
Tornillo y tuercas	20	200	4.000
TOTAL, Materiales			634.890
Mano de obra en elaboración de los modelos	25 horas	25.000	625.000
TOTAL, Mano de obra y Materiales			1.259.890

5.3 Presupuesto para el simulador de vuelo.

Con respecto a la anterior tabla, se puede observar claramente que un simulador con estas 3 componentes (básicas para el proceso de aprendizaje de un piloto) se puede construir por menos de 2.000.000 millones de pesos, cabe aclarar que falta sumarle el desarrollo e investigación en el código de programación que podría ser un porcentaje de un valor establecido por el desarrollador. A diferencia de las pocas consolas y módulos que se

consiguen en el mercado, que no deja de ser simples juguetes, con esta investigación y desarrollo se puede obtener una solución de categoría B a un costo muy bajo, ya en trabajos futuros se expondrán los ítems faltantes para considerar a este simulador de categoría B, realizando un debido estudio de mercadeo para sacar un promedio de materiales a utilizar respecto a costos.

CONCLUSIONES

- La conclusión más importante con respecto al simulador de vuelo es su funcionamiento probado por un piloto. Unas de las tantas pruebas realizadas consistían en un aterrizaje de la aeronave completamente desalineada a la pista, recuperando su posición con un buen aterrizaje, lo que demuestra que la interfaz facilita hacer maniobras complicadas en pequeños lapsos de tiempo, esto muestra que el sistema proporciona fiabilidad.

- Para vuelos por instrumentos, el simulador proporciona facilidad en sintonizar las coordenadas y comunicaciones sin perder contacto con el estado de la aeronave, simulando lo que haría una interacción real torre de control – aeronave y aeronave – torre de control.

- El lenguaje de programación implementado para el desarrollo de la interfaz entre el simulador y los componentes electrónicos (gpios de la board) es Python, ya que capacidad de procesamiento del microprocesador es lo suficientemente alta para garantizar latencias bajas para la simulación de las interfaces y de esta manera aprovechar la gran cantidad de librerías y herramienta que proporciona dicho lenguaje.

- El uso de la interfaz LINK2FS en algunos casos no proporciona el control total al programador en la extracción de datos, lo que se compensa con modificaciones de algunos datos para la visualización de estos.

- La board de desarrollo Beaglebone black con respecto al rendimiento y funcionamiento en el tráfico de datos funciona bien, la única desventaja es que muchos de los pines están reservados, teniendo que acudir a otra tarjeta de desarrollo para suplir la demanda de pines.

- La construcción del IDE de desarrollo proporciona todas las herramientas a un programador de las board de desarrollo para la familia STM32XXX con un único problema, en caso de haber errores en programación o compilación hay poca información en la web de como reparar dichos daños.

TRABAJOS FUTUROS

- Construcción de una tarjeta electrónica propia que incluya todos los componentes electrónicos, en especial el microprocesador y microcontrolador.

- Diseño y construcción de un timón con fuerza (Feedback force) para el control de la

- Diseño y construcción de la cabina a escala de la aeronave con sus respectivos componentes.

- Desarrollo de software propio para la digitalización de los instrumentos en pantalla.

- Desarrollo de software propio para la interfaz de datos enviados y recibidos de cualquier componente electrónico utilizando el SDK que proporciona Microsoft para la construcción de la interfaz.

BIBLIOGRAFÍA

[1] ASME, The Link Flight Trainer, Nueva York. EEUU: Robertson Museum and Science Center, 2010.

[2] FAA, «ELECTRONIC CODE OF FEDERAL REGULATIONS Part 121,» U.S. Goverment Publishing Office, 1 Mayo 2017. [En línea]. Available: https://www.ecfr.gov/cgi-bin/text-idx?tpl=/ecfrbrowse/Title14/14cfr121_main_02.tpl. [Último acceso: 2 Mayo 2017].

[3] S. J. Y. Ashok Kuppusamy, «Design of Reversible Control Loading System for a Fixed Wing Aircraft Using XPlane,» de *Mechanical and Aerospace Engineering (ICMAE)*, Londres, 2016.

[4] Chris Binns Egnatia Aviation Chrysoupolis, Greece, *Aircraft Systems Instruments, Communications, Navigation, and Control*, © 2019 John Wiley & Son

[5] C. R. Spitzer, Avionics development and implementation, Williamsburg, Virginia, U.S.A: CRC press, 2007.

[6] Raymer, D. P. (2012). *Aircraft Design A Conceptual Appoach.* AIAA Education Series.

[7] Corke, T. C. (2002). *Design of Aircraft* . Prentice Hall.

[8] Thom, T. (2013). *Human Factors & pilot perfomance.* Pooley's air pilot Publishing.

[9] Fente, P. R. (2018, Junio 14). *Noticias Aereas*. Retrieved from https://noticiasaereas.com/simuladores-vuelo-airbus-stc/

[10] Thom, T. (2012). *Radio Navigation & intrument flying.* Pooley's air pilot publishing.

[11] pallet, E. H. (2009). *Aircraft instruments.* Pearson .

[12] FAA, F. A. (2013). *Instrument Flying Handbook.* Newcastle, United States: Aviation Supplies & Academics Inc.

[13] Thom, T. (2012). *the aeroplane technical.* England: pooley's air pilot publishing .

[14] Sychev, V. (2012). *Baron 58 Copkit Simulator*. Retrieved from http://www.simvim.com/b58/

[15] Microsoft, S. t. (2017, febrero 28). *Microsoft.* Retrieved from https://support.microsoft.com/es-es/help/925724/flight-simulator-x-minimum-system-requirements

[16] beagleBone. (2018, Junio 28). *BeagleBone.* Retrieved from https://beagleboard.org/black

[17] Mbed, A. (2018). *St.* Retrieved from https://www.st.com/en/evaluation-tools/nucleo-f401re.html

[18] Integrated, M. (2003). *datasheets maximintegrated.* Retrieved from https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf

[19] CO, X. A. (2008, octubre 29). *SparkFun.* Retrieved from https://www.sparkfun.com/datasheets/LCD/ADM1602K-NSW-FBS-3.3v.pdf

[20] Bench, H. (2015, Junio 16). *eeshop.* Retrieved from http://eeshop.unl.edu/pdf/KEYES%20Rotary%20encoder%20module%20KY-040.pdf

[21] Wikipedia, F. (2018, Mayo 26). *Wikipedia.* Retrieved from https://es.wikipedia.org/wiki/Interruptor

[22] Wikipedia, F. (2018, Octubre 24). *Wikipedia.* Retrieved from https://es.wikipedia.org/wiki/Potenci%C3%B3metro

[23] developers, T. p. (2018). *pypi.* Retrieved from https://pypi.org/project/pip/

[24] Caudle, S. (2018). *pypi.* Retrieved from https://pypi.org/project/spidev/

[25] Hull, R. (2018). *pypi.* Retrieved from https://pypi.org/project/max7219/

[26] Liechti, C. (2001 - 2016). *pySerial.* Retrieved from https://pyserial.readthedocs.io/en/latest/pyserial.html

[27] DiCola, T. (2018). *pypi.* Retrieved from https://pypi.org/project/Adafruit-GPIO/

[28] Kościów, B. (2018). *pypi.* Retrieved from https://pypi.org/project/CharLCD/

[29] 93, m. t. (2016). *hotboards.* Retrieved from http://hotboards.org/index.php/es/blog/20-spanish/blog/st/102-programando-con-el-bootloader

[30] *An half.* (2015, Marzo 7). Retrieved from https://www.carminenoviello.com/2014/12/28/setting-gcceclipse-toolchain-stm32nucleo-part-1/

[31] Ionescu, L. (2018, Abril 25). *GNU MCU ECLIPSE.* Retrieved from https://gnu-mcu-eclipse.github.io/tutorials/blinky-arm/

ANEXOS

A- ANEXO DE LA CONFIGURACION DEL IDE DE DESARROLLO

Para la configuración del IDE se decide utilizar la plataforma de software Eclipse, que posee un conjunto de herramientas de programación y de código abierto para desarrollar este proyecto. Primero se debe descargar el software del siguiente enlace: *https://www.eclipse.org/downloads/*, preferiblemente la versión para desarrolladores en lenguaje C/ C++. Se debe descargar de la página oficial de ST los drivers para controlar el puerto serial de la tarjeta de desarrollo para la comunicación vía USB, esos drivers se descargan del siguiente link: https://www.st.com/content/st_com/en/products/development-tools/software-development-tools/stm32-software-development-tools/stm32-utilities/stsw-link009.html [30]

Además del IDE de desarrollo se necesitan algunas herramientas para poder compilar o construir el proyecto, generar un archivo binario, programar la tarjeta y depurar, para ello se debe tener instalado un toolchain, el cual es un conjunto de proyectos que contiene herramientas de desarrollo de software producidas por el proyecto de GNU (sistema operativo tipo unix), para este caso se utiliza GCC ARM Embedded, que posee herramientas para diferentes procesadores de la familia Cortex-M y se puede descargar del siguiente link: https://launchpad.net/gcc-arm-embedded, preferiblemente descargar la versión .zip.

Una vez descargado, se prueba el toolchain, esto se realiza desde la consola CMD de Windows, entrando al directorio del archivo descargado y ejecutando la versión del sistema, se realiza con el siguiente comando: ARMTC/bin /arm-none-eabi-gcc -v. Al trabajar con dependencias de tipo GNU en Windows se deben descargar algunas funciones originales del sistema unix para poder ejecutar el toolchain sin ningún problema. Es necesario descargar los binarios de las dependencias necesarias, como es el caso de make (compilar archivos), echo (imprimir texto en pantalla), rm (borrar archivos o directorios), touch (crear archivos) y uname (imprimir información del sistema); este conjunto de archivos se pueden descargar de los siguientes enlaces http://gnuwin32.sourceforge.net/packages/make.htm, http://gnuwin32.sourceforge.net/packages/coreutils.htm y deben ser extraídos en la siguiente ruta ARMTC/bin.

Hasta este punto ya se tiene parte de las herramientas para poder trabajar con el procesador de la tarjeta de desarrollo, paso siguiente, se deben instalar algunas extensiones para que eclipse habilite las opciones de compilación, programación y debbugin. Para ello en la siguiente ruta dentro de eclipse help > install new software, se descarga los plugin necesarios con la siguiente url: http://gnuarmeclipse.sourceforge.net/updates.

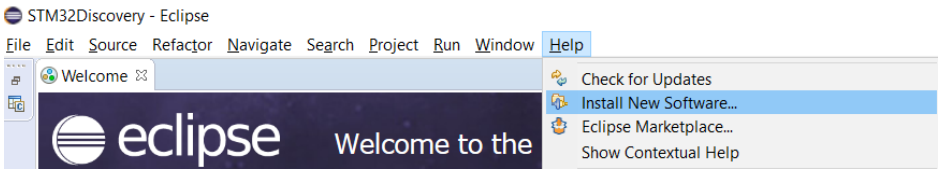

Figura 4.1 menú de nuevos paquetes.

Se habilitan solamente las siguientes opciones, select cross compiler, stm32fx Project, j-link debuggin, openOCD debuggin.

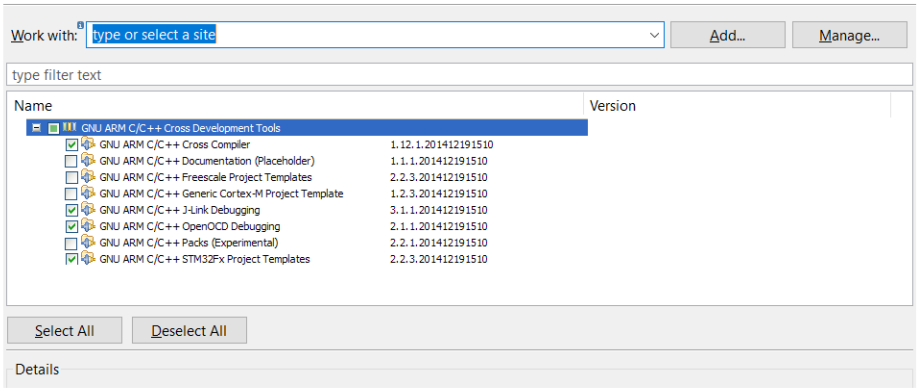

Figura 4.2 paquetes de herramientas de compilación y librerías.

En caso de que estos ítems no aparezcan, se debe instalar una dependencia en la versión 8.5 por medio del siguiente link: http://download.eclipse.org/tools/cdt/releases/8.5, y se habilita solamente C/C++ development tools sdk.

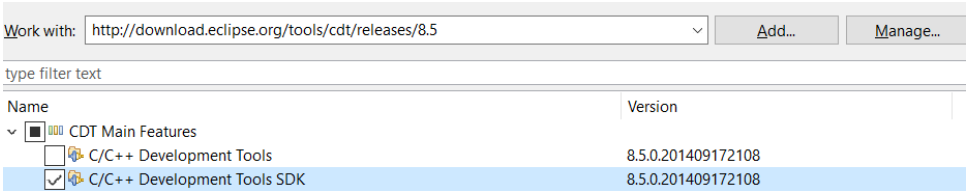

Figura 4.3 paquete de herramientas del SDK.

Ahora para hacer debuggin en tiempo real, para la tarjeta de desarrollo stm32f4 nucle, se hace uso de la herramienta openOCD versión 0.9.0 que se descarga del siguiente link: http://www.freddiechopin.info/en/download/category/4-openocd. Esta herramienta internamante tiene pregconfigura diferntes instrucciones y procesadores permitiendo hacer la depuración del código con el cual se esté trabajando. Para probar la conexión entre la tarjeta y el software por medio del CMD, dentro de la carpeta bin, se utiliza el siguiente comando openocd -f board/stm32f4nucleo.cfg

Para poder realizar la depuración de cualquier código con la tarjeta de desarrollo, se debe llamar el software de openOCD de la siguiente forma en la ruta: window>preference>run/debug>string substitution y se agregan dos nuevas variables, la ruta de la carpeta bin (value: ruta de la carpeta bin, new variable: openocd_path) y el ejecutable del openoCD (value: openocd.exe, new variable: openocd_executable)

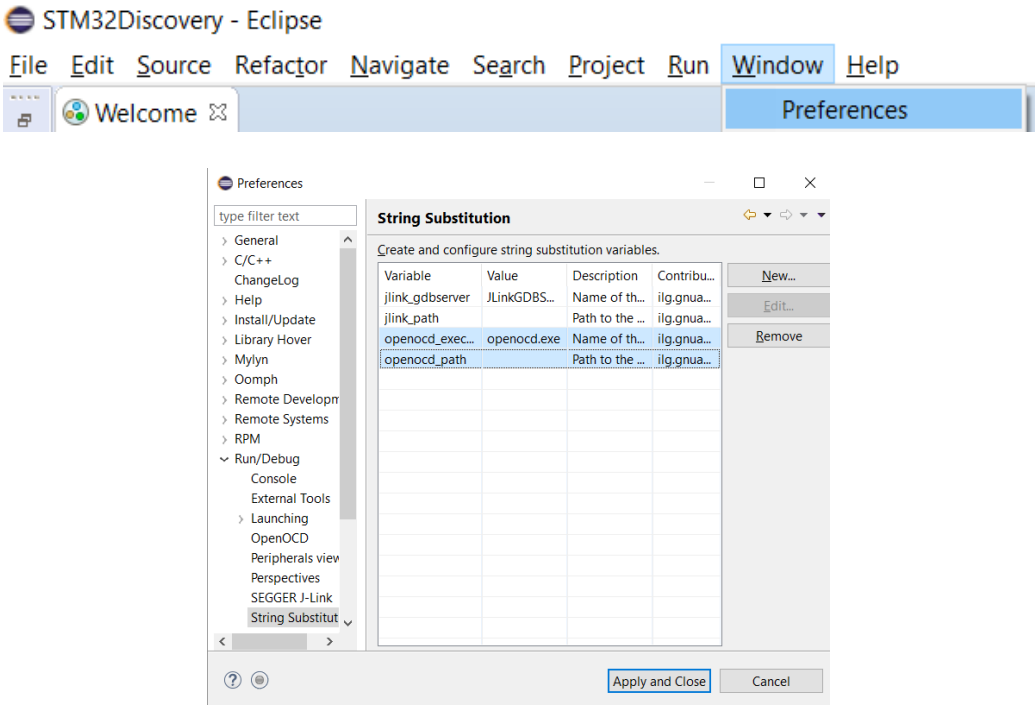

Figura 4.4 componentes de depuración.

Por último, para que eclipse reconozca las acciones que debe ejecutar con GCC ARM Embedded y openOCD para programar la tarjeta, se debe agregar los siguientes comandos en: run>external tools >external tools configuration>new configuration.

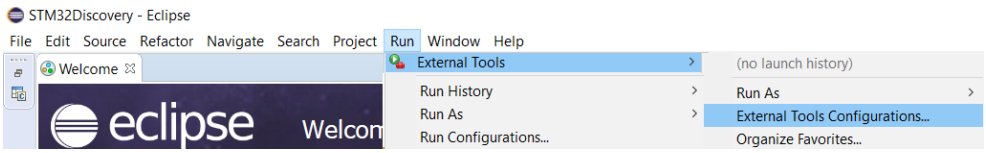

Figura 4.5 menú herramientas de compilación.

- name: flash_nucleof4.
- location: ${openocd_path}\${openocd_executable}
- working directory: ${project_loc}
- arguments: -f board/stm32f4discovery.cfg
  -c “program Debug/${project_name}.hex verify reset”

-c exit

Con respecto a los argumentos -f, busca el directorio y -c, los ejecuta.

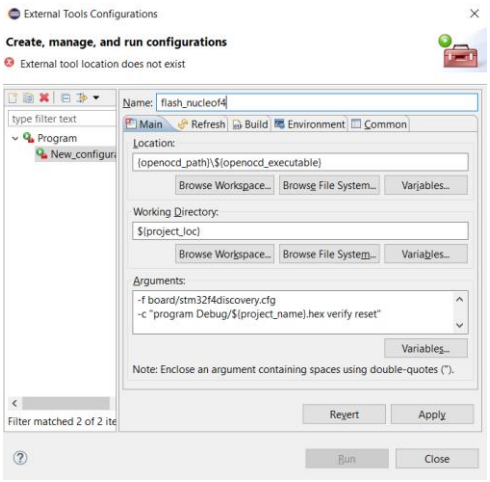

Figura 4.6 parámetros de compilación.

Ahora, para depurar el código con la tarjeta en tiempo real se debe agregar los siguientes comandos en: run>debug configurations > GDB OpenOCD Debugging >new configuration

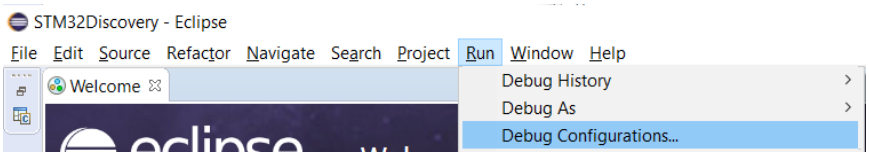

Figura 4.7 menú de herramientas de depuración.

Name: template_nucleoF4
Main:
- project: ${project_loc}
- C/C++ Application: buscar proyecto, seleccionar archivo .elf.
- build before launchig: seleccionar disable auto built.

Debugger:
- Executable: ${openocd_path}/${openocd_executable}
- Config options: -f board/stm32f4discovery.cfg
  reset_config srst_only srst_nogate

Startup:
- Initialization commands: desmarcar Enable ARM semihosting.

Common:
- Display in favorites menu: seleccionar debug.

A.1 Configuración de librerías para la tarjeta de desarrollo stm32f4 nucleo

Después de haber configura el IDE de desarrollo, se configura un template base para que cada proyecto arranque con las librerías ya pre-cargadas, esto se realiza gracias a los plugin instalados previamente. Para crear un nuevo proyecto se sigue la siguiente ruta: file > new > C++ project:

- project name: nombre que se le va a dar al proyecto.
- project type: ST32F4xx C/C++ project.
- Toolchains: Cross ARM GCC.
- Target processor settings: solo se debe modificar el chip family, flash size y external clock(Hz) con respecto al que se vaya a trabajar, estos datos se pueden encontrar en el datasheet del microcontrolador, el resto de opciones no se modifican.

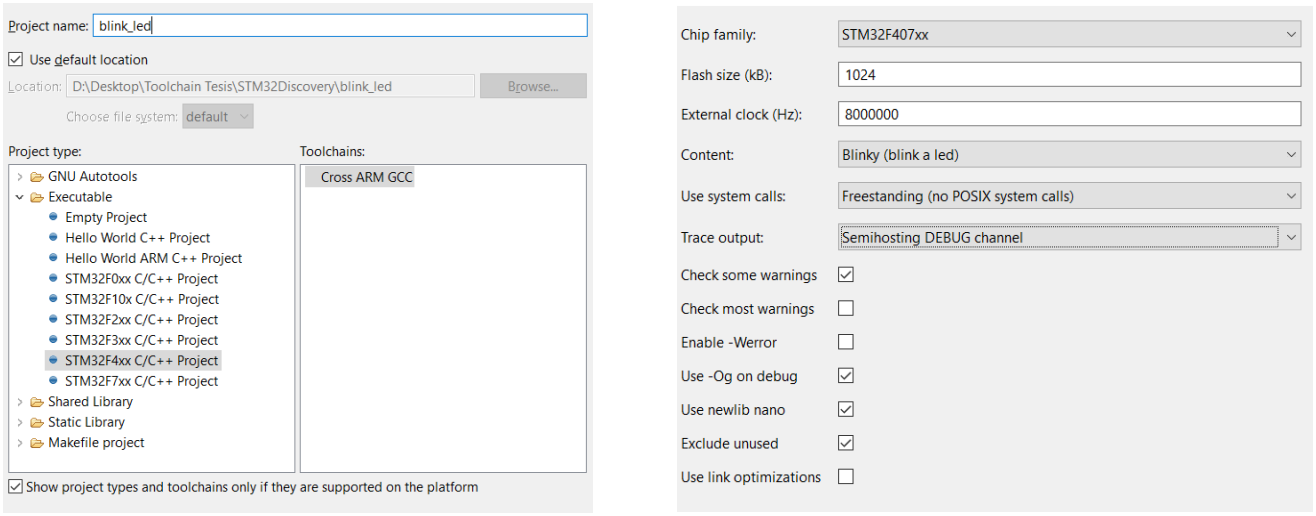

Figura 4.8 template base para cualquier ejemplo.

- Folders Settings: estos parámetros se dejan por defecto, incluyen las librerías y carpetas necesarias para cualquier proyecto.
- Select configurations: por defecto estarán seleccionados Debug, Release, estas serán las 2 acciones que permite el software eclipse ejecutar para el microcontrolador a programar. [31]

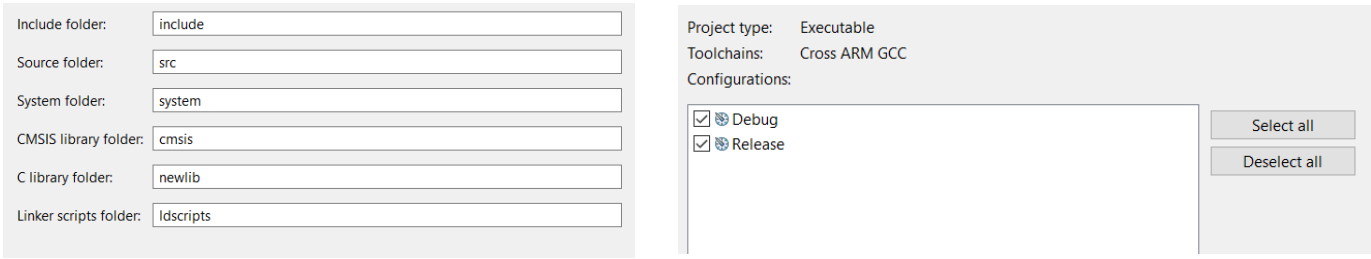

Figura 4.9 template base para cualquier ejemplo.

A este punto de la investigación ya se tienen configuras las 2 tarjetas de desarrollo, con cualquiera de las 2 se puede trabajar en cualquier aplicación, para este caso, el simulador de vuelo se utiliza la Beaglebone como sistema principal que pueda ejecutar varias tareas a la vez, comunicarse con el software de interfaz (link2fs) del fligth simulator X vía serial, y tener control sobre algunos periféricos como es el caso de los encoders rotativos, las pantallas 7 segmentos que se comunican vía SPI, un despliegue lcd de 16x2 y algunos botones. Por otro lado, la tarjeta stm32f4 Núcleo se encarga también de la comunicación serial con el software de interfaz y controlar algunos botones y 2 ADC. De esta manera ya se tiene repartidas las funciones de cada componente. En la próxima sección se habla un poco del software link2fs que es el encargado de interfaz el simulador con las tarjetas de desarrollo.

B- Anexo de los scripts de programación del código de cada componente.

B1- Implementación del filtro de los parámetros de salida.

Los parámetros de salida de link2fs para los visualizadores de la cabina vienen acompañados por una clave que el software entrega, dicha clave puede ser del estilo “=x”, “<x”, “?x”, “$x” donde x es la letra que representa el valor a mostrar en el indicador, entonces se debe implementar una especie de filtro vía comunicación serial que es por donde va recibir la información, el cual clasifique la cable y la letra a la que corresponde y a su vez asignarle el tamaño de byte a recibir. Todos los códigos de los scripts para la beaglebone black están hechos en Python con el fin de optimizar el código con las librerías y las dependencias anteriormente instaladas.

```
import Adafruit_BBIO.UART as UART
#se llama la librería de adafruit para controlar el UART con la sintaxis que ellos proponen.
import serial
#importa la librería serial para comunicación.
import os
#importa funciones para asignar datos al sistema operativo.
UART.setup("UART1”)
#se define uno de los 5 puertos seriales que tiene la tarjeta de desarrollo para la comunican
#con link2fs.
os.system("config-pin " + "P9_24" + " uart")
os.system("config-pin " + "P9_26" + " uart")
#se le indica al sistema operativo que los pines, serán utilizados como puerto serial de
#transmisión y recepción de datos.
ser = serial.Serial("/dev/ttyO1", baudrate=115200)
#se declara una variable para poder ejecutar los comandos vía serial, además se le da el
#nombre del puerto serial que debian le asigna a la tarjeta de desarrollo con su respetiva
#velocidad de datos.
ser.open()
#se abre el Puerto serial.
if ser.isOpen():
#si el Puerto está abierto ejecutar la siguiente línea de código.
   while(True):
#realizar el ciclo mientras haya información en el puerto serial.
      input = ser.readline()
#se guarda la información obtenida del puerto serial en una variable.
      if(len(input.split("=")) > 1):
#En esta linea de codigo, se ejecutan 2 opciones dentro del ciclo if, primero por medio de
#split, se busca si la clave es un ‘igual’, después se saca a longitud de cuantos claves encontró,
```

```
#para este caso solo va encontrar uno, por ende, el condicional dice que si hay mas de uno
#ejecute las siguientes líneas de código.
        input = input.replace('\r\n','')
#ahora la en la variable entrada donde hay salto de línea se reemplaza por un espacio vacío.
        input = input.split('=')
#se vuelve a separar donde esta la clave '=' pero esta vez para leer el indicador que sigue.
        print input
#se imprime la información que esta después de la clave.
        for x in range(len(input)):
#se obtiene la longitud de input por medio de la función len, y el resultado sera la cantidad
#del rango donde tiene que iterar el ciclo.
            if(input[x].find('A') == 0):
#por medio de la función find se encuentra un parámetro, en este caso será A para los
#visualizadores de los radios de comunicación.
             input[x] = input[x].replace("A", '')
#se reemplaza A, por un valor nulo vacío con el fin de seguir leyendo el valor que esta
#enviando en este comando
ser.close()
#Cierra la comunicación serial.
```

Por medio de este template se puede obtener el valor que está enviando cada indicador y ya asígnale el dispositivo electrónico por el cual va a desplegar dicha información.3.5.

B-2 Script de funcionamiento de los despliegues de 7 segmentos.

Aprovechando las librerías posteriormente instaladas, se puede visualizar cualquier conjunto numérico de hasta 8 datos con decimales si se desea, para trabajar directamente con los radios de comunicación, navegación y transponder. Las librerías realmente ayudan a que en pocas líneas haya un perfecto funcionamiento sobre los despliegues de 7 segmentos, como lo muestra el siguiente ejemplo:

```
import max7219.led as led
# importa la librería max7219, es la que tiene cada uno de los códigos que maneja el integrado
#que internamente tiene el módulo de 7 segmentos, convierte las instrucciones que recibe por
#spi en registro que maneja dicho integra para desplegar cualquier valor numérico.
import spidev
#importa la librería de Linux para controlar dispositivos spi.
import os
#importa funciones para asignar datos al sistema operativo.
from math import floor
#importa la función de aproximación a entero de un valor decimal al numero que esta por
#debajo.
SPIBUS = 1
#puerto spi a utilizar, en el caso de la Beaglebone, puede ser la opción 1 o 2.
SPIDEV = 0
#esta es la dirección que contiene el SPIBUS 1.
os.system("config-pin P9.17 spi")
```

```
os.system("config-pin P9.18 spi")
os.system("config-pin P9.21 spi")
os.system("config-pin P9.22 spi")
#la comunicación SPI se realiza entre 2 o más dispositivos, debe haber un maestro que es el encargado de recibir los datos y uno o mas esclavos que son los encargados de recolectar la información, esta comunicación se realiza a través de 4 pines: clock (marca la sincronización), mosi (salidas de datos del master), miso (salida de datos del esclavo) y chipselect (selecciona el esclavo con el que va a realizar la comunicación), estos pines se asignan atreves de funciones del sistema

ssg=led.sevensegment(cascaded=5, spi_bus=SPIBUS, spi_device=SPIDEV)
#por medio de la función led.sevensegment se le indica al script cuantos dispositivos se
#conectaran en cascada para su funcionamiento y con que puerto spi se va a trabajar
spi = spidev.SpiDev()
#activa la comunicación spi por parte del sistema operativo.
spi.open(SPIBUS,SPIDEV)
#abre el puerto para inicializar la comunicación
spi.max_speed_hz = 200000
#estable la velocidad de comunicación de los dispositivos, es importante tener clara la
#velocidad ya que si es errónea solo se verá ruido en los despliegues.
spi.bits_per_word = 0x08
#soporta por numero o letra solo 8 bits de información en la comunicación con el sistema
#operativo.

#modo de operación entre los dispositivos externos y el sistema operativo, a través de este
#modo se configura la polaridad de reloj y fase.
ssg.brightness(8)
#controla la luminosidad de cada despliegue, los valores van desde 1 hasta 15.
ssg.write_number(deviceId=4,value=-0.001+float(input[x].replace(',','.')), decimalPlaces=2)
#escribe un valor sobre el 5 despliegue, también cambia el valor flotante que recibe del
#software link2fs por un valor decimal que es modificado por un valor que suma para
#compensar el offset, además se le indica al código que debe reservar los 2 últimos dos
#dígitos como decimales.
```

B-3 Script de funcionamiento de la pantalla LCD de 16x2

Para la pantalla LCD se hace uso de las librerías ya instaladas que contienen la codificación de cada carácter y poder visualizar los datos sin ningún error, esta pantalla es utilizada para desplegar toda la información del piloto automático, la configuración que se debe tener en cuenta para el script se muestra el siguiente ejemplo:

```
import Adafruit_CharLCD as LCD
#importa la libreria bajo los parámetros de la sintaxis de adafruit para poder controlar la
#pantalla LCD.
import Adafruit_BBIO.GPIO as GPIO
#importa la librería bajo los parámetros de la sintaxis de adafruit para poder controlar
#cualquier pin GPIO.
```

```
lcd_rs        = "P8_13"
lcd_en        = "P8_14"
lcd_d4        = "P8_15"
lcd_d5        = "P8_16"
lcd_d6        = "P8_17"
lcd_d7        = "P8_18"
#se asigna cada uno de los pines de la pantalla a pines de la tarjeta de desarrollo beaglebone.
os.system("config-pin " + lcd_rs + " gpio")
os.system("config-pin " + lcd_en + " gpio")
os.system("config-pin " + lcd_d4 + " gpio")
os.system("config-pin " + lcd_d5 + " gpio")
os.system("config-pin " + lcd_d6 + " gpio")
os.system("config-pin " + lcd_d7 + " gpio")
#se le indica al sistema operativo que los pines ya definidos estarán como GPIOS y pueden
#ser utilziados como entradas o salidas.
GPIO.setup(lcd_rs, GPIO.OUT)
GPIO.setup(lcd_en, GPIO.OUT)
GPIO.setup(lcd_d4, GPIO.OUT)
GPIO.setup(lcd_d5, GPIO.OUT)
GPIO.setup(lcd_d6, GPIO.OUT)
GPIO.setup(lcd_d7, GPIO.OUT)
#todos los pines de la pantalla LCD envían datos para la configuración de esta, por ende,
#cada pin debe ser declaro como salida.
lcd_columns = 16
#al ser una pantalla de 16x2 traduce que tiene 16 columnas que internamente esta compuesta
#por 40 bits para despliegue de cualquier información.
lcd_rows    = 2
#al ser una pantalla de 16x2 traduce que tiene 2 filas para desplegar la información.
lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6, lcd_d7,
                           lcd_columns, lcd_rows)
#la function LCD.Adafruit_CharLCD asigna a los pines ya establecidos y las dimensiones
#como parámetros de configuración.
lcd.clear()
#limpia cualquier dato que haya en la pantalla.
lcd_line1_alt = ""
lcd_line1 = "   "
lcd_line2 = "   "
lcd_line2_vs = ""
#se definen algunas variables de texto vacía para formatear más adelante la información.
lcd_update = False
#se define la variable lcd_update para un condicional más adelante, que en caso de recibir
#los datos correctos, ejecute la acción de imprimir información en la pantalla LCD.
#En este punto no se tendrá en cuenta le sistema de filtro, aunque es necesario para la
#recepción de datos del piloto automático, pero ya se explicó anteriormente en este capítulo.
if(input[x].find('b') == 0):
        input[x] = input[x].replace("b", '')
        lcd_line1_alt = " ALT " + input[x]
```

```
#En caso de que la clave recibida en el puerto serial sea igual a b, esta tendrá asociada un valor que es altura programa en el piloto automático, de esta manera también se implementa el filtro para la velocidad vertical, la activación del HDG, NAV y APR.
        lcd_update = True
#al entrar a este condicional y comprobar que hay datos, la variable LCD_update será #verdadera y el script procede a ejecutar la función de imprimir los datos recibidos en la #pantalla.


if(lcd_update):
#si la variable lcd_update es verdadera, es que hay información para imprimir nuevos datos #en la pantalla LCD.
        lcd.clear()
#limpia los datos que hay en pantalla.
        lcd.message(lcd_line1 + lcd_line1_alt + "\n" + lcd_line2 + lcd_line2_vs)
#envía un mensaje a la pantalla con los nuevos datos obtenidos.
        lcd_update = False
#por último vuelve de nuevo la variable lcd_update falsa con el fin de poner de nuevo el filtro #en funcionamiento.
```

B-4 Script de funcionamiento de los Encoders Rotativos.

Los encoders rotativos son utilizados para generar cambios en los valores de las frecuencias normales y stanby de los dispositivos de comunicación y navegación, esto lo hacen ya que cada cambio es representado por un giro que a su vez es transformado en un pulso, un pin controla los pulsos mientras que otro el giro si es a la izquierda o derecha. Para su funcionamiento se han implementado interrupciones en cada pin declarado para estos dispositivos, ya que es necesario un tiempo de espera para lectura de datos y no vale la pena parar todo el sistema utilizando funciones como delay ciertos milisegundos ya que el filtro y el puerto serial todo el tiempo tienen que estar a la escucha, por ende, se ha optado por utilizar interrupciones, la configuración que se debe tener en cuenta para el script se muestra el siguiente ejemplo:

```
import Adafruit_BBIO.GPIO as GPIO
#importa la librería bajo los parámetros de la sintaxis de adafruit para poder controlar #cualquier pin GPIO.
import os
#importa funciones para asignar datos al sistema operativo.
Enc1_Clk = "P9_11"
#define el pin para lectura de pulso.
Enc1_Dt   = "P9_13"
#define el pin para detección de giro.
os.system("config-pin " + Enc1_Clk + " gpio")
os.system("config-pin " + Enc1_Dt + " gpio")
#se le indica al sistema operativo que los pines ya definidos estarán como GPIOS y pueden #ser utilizados como entradas o salidas.
GPIO.setup(Enc1_Clk, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```

```
# se declara el pin de lectura de pulso como entrada y con interrupciones, pull_up_down
#significa que estar pendiente de alguna transición de alto a bajo.
GPIO.setup(Enc1_Dt, GPIO.IN, pull_up_down = GPIO.PUD_UP)
# se declara el pin de detección de giro como entrada y con interrupciones, pull_up_down
#signfica que estar pendiente de alguna transición de alto a bajo.

Contador_1 = 0
#se crea una variable para contar el número de pasos.
Rotacion_1 = True
#se crea una variable para indicar el sentido
Enc1_Clk_Ultima = 0
#variable para guardar información de la última lectura.
Enc1_Clk_Actual = 0
#variable para guardar información de la lectura actual.
Enc1_Clk_Ultima = GPIO.input(Enc1_Clk)
#Guarda en la variable Enc1_Clk_Ultima el valor leído en el pin Enc1_Clk.
def detectionEnc1(null):
#se define la función que va calcular el número de pasos y el giro.
   Enc1_Clk_Actual = GPIO.input(Enc1_Clk)
#se vuelve a leer el pin Enc1_Clk por si ha habido cambio desde la anterior lectura.
   if Enc1_Clk_Actual != enc1_clk_ultima:
#si la lectura anterior es diferente a la actual ejecuta el siguiente condicional.
      if GPIO.input(Enc1_Dt) != Enc1_Clk_Actual:
#si hay cambio de giro, es diferente al valor de Enc1_Clk_Actual y por ende se cumple la
#siguiente condicional para identificar el giro.
         Rotacion_1 = True
#giro hacia la derecha.
      else:
#si los valores son iguales.
         Rotacion_1 = False
#el giro es hacia la izquierda.
      if Rotacion_1:
#si la rotación es verdadera, se enviará por el puerto serial la siguiente clave para generar
#cambios en el la frecuencia de comunicación, aumentando el valor en el indicador.
            ser.write("A04 \n")
      else:
#si la rotación es falsa, disminuirá el valor en los radios.
            ser.write("A03 \n")
GPIO.add_event_detect(Enc1_Clk,GPIO.BOTH,callback=detectionEnc1,bouncetime=10)
#este comando se dispara en caso de detectar algún evento de transición de bajo a alto o de
#alto a bajo en el pin Enc1_Clk, si detecta dicho cambio, llama a la función establecida como
#detectionEnc1 y con un tiempo de espera de 10 ms.
```

B-5 Script de funcionamiento de los pulsadores.

Con respecto al script de los botones, posiblemente sea el código más simple para implementar, cada vez que se pulse alguno, cambiara su estado de 0 a 1 o de 1 a 0

dependiendo de la polarización que tenga con la resistencia de carga, la idea es proporcionar precisión y realismo dentro del simulador, pero hay un problema, cuando se utiliza varias veces el mismo botón, aparece pulsos extras, cambiando totalmente el dato con el cual se está trabajando, existen varias razones por la cuales sucede esto. La primer causa es debido a que el código está en un ciclo que se repite miles de veces por segundo, esto provoca que cuando se pulsa un botón una sola vez, la tarjeta de desarrollo en algunas ocasiones leerá un cierto número más de veces, esto se puede solucionar trabajando con interrupciones del microprocesador, de la misma forma como se ha trabajado los encoders rotativos, cuando haya una transición de estado, este dispare una función, controlando por x tiempo en milisegundos la lectura, dando así una ventana de tiempo en un próximo cambio de estado. La segunda causa del problema de rebote es debido a la propia construcción del botón. Un botón internamente, no es más que 2 láminas metálicas que se unen o se separan por medio de un resorte. En el momento de la unión o espacio de las láminas, el resorte provoca una serie de rebotes entre las láminas que el microprocesador es capaz de detectar. La solución para este inconveniente, es que cada vez que haya un evento relacionado con los botones se dispare cuando se suelte el botón y no cuando se presione, de esta forma combinado con las interrupciones, se puede tener buenos resultados, la configuración que se debe tener en cuenta para el script se muestra el siguiente ejemplo:

```
import Adafruit_BBIO.GPIO as GPIO
#importa la librería bajo los parámetros de la sintaxis de adafruit para poder controlar
#cualquier pin GPIO.
import os
#importa funciones para asignar datos al sistema operativo.
os.system("config-pin " + BTN1 + " gpio")
os.system("config-pin " + BTN2 + " gpio")
os.system("config-pin " + BTN3 + " gpio")
#se le indica al sistema operativo que los pines ya definidos estarán como GPIOS y pueden
#ser utilizados como entradas o salidas.
BTN1 = "P9_41"
BTN2 = "P9_42"
BTN3 = "P9_30"
#se asignan pines de las tarjetas de desarrollo para los botones.
GPIO.setup(BTN1, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(BTN2, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(BTN3, GPIO.IN, pull_up_down = GPIO.PUD_UP)
# se declara los pines de cambio de estado como entrada y con interrupciones, pull_up_down
def detectionBtn1(null):
#se define una funcion para cada uno de los botones, cuando sean accionador mande por
#medio del protocolo serial el valor a modificar.
        ser.write("A06 \n")
def detectionBtn2(null):
        ser.write("A18 \n")
def detectionBtn3(null):
        ser.write("A48 \n")
GPIO.add_event_detect(BTN1, GPIO.RISING, callback=detectionBtn1, bouncetime=10)
GPIO.add_event_detect(BTN2, GPIO.RISING, callback=detectionBtn2, bouncetime=10)
```

```
GPIO.add_event_detect(BTN3, GPIO.RISING, callback=detectionBtn3, bouncetime=10)
#este comando se dispara en caso de detectar algún evento de transición de bajo a alto en el
#botón declarado, si detecta dicho cambio, llama a la función establecida como detectionBtn1
#y con un tiempo de espera de 10 ms.
```

B-6 Script de funcionamiento de los Leds

Este script nace con el fin de suplir el funcionamiento mecánico que tiene internamente los botones del piloto automático, en caso de presionar un botón y que este ejerza alguna acción sobre otro, físicamente no están conectados entre ellos para ejecutar una acción sobre otro, pero con la lógica de programación se puede suplir esta necesidad y transferir este tipo de mecanismo a un conjunto de luces emitidas por un grupo de leds y construir un simulador un más real, la configuración que se debe tener en cuenta para el script se muestra el siguiente ejemplo:

```
import Adafruit_BBIO.GPIO as GPIO
#importa la librería bajo los parámetros de la sintaxis de adafruit para poder controlar
#cualquier pin GPIO.
import os
#importa funciones para asignar datos al sistema operativo.
ledAPR = "P8_9"
ledNAV = "P8_10"
ledHDG = "P8_11"
ledAP = "P8_12"
ledREV = "P8_46"
ledALT = "P8_44"
#se declaran 6 pines para los leds que estaran asociados a la lectura de los botones del piloto
#automatico
os.system("config-pin " + ledHDG + " gpio")
os.system("config-pin " + ledAP + " gpio")
os.system("config-pin " + ledNAV + " gpio")
os.system("config-pin " + ledAPR + " gpio")
os.system("config-pin " + ledREV + " gpio")
os.system("config-pin " + ledALT + " gpio")
#se le indica al sistema operativo que los pines ya definidos estarán como GPIOS y pueden
#ser utilizados como entradas o salidas.
GPIO.setup(ledHDG, GPIO.OUT)
GPIO.setup(ledAP, GPIO.OUT)
GPIO.setup(ledNAV, GPIO.OUT)
GPIO.setup(ledAPR, GPIO.OUT)
GPIO.setup(ledREV, GPIO.OUT)
GPIO.setup(ledALT, GPIO.OUT)
#se declaran los 6 pines como salida
if(input[x].find('n') == 0):
#implementado el filtro que se hablo en la primera sección de este capítulo, se trabaja con las
#claves de los botones del piloto automático, y se maneja una lógica para encender dichos
#leds.
```

```
        input[x] = input[x].replace("n", '')
#lee la clave con el valor asociado al boto REV
        if(input[x] == '1'):
#si dicho valor es alto, enciende el led que le corresponde al botón
                GPIO.output(ledREV,GPIO.HIGH)
        else:
#de lo contrario si registra algún cambio en la llave 'n' accionada por otro botón del piloto
#automático, este se apagara de inmediato
                GPIO.output(ledREV,GPIO.LOW)
        if(input[x].find('k') == 0):
#de esta forma se organizan los 6 leds para cada uno de los botones, y está representado en
#este script solamente para 2, con el fin de no alargar el texto.
                input[x] = input[x].replace("k", '')
                        if(input[x] == '1'):
                                GPIO.output(ledALT,GPIO.HIGH)
                        else:
                                GPIO.output(ledALT,GPIO.LOW)
```

B-7 Script de configuración de la tarjeta STM32F4 núcleo.

Debido al problema que hay por falta de pines de la beaglebone con respecto a la cantidad de componentes electrónicos, se ha decidido implantar un script que controle los periféricos faltantes por medio del microcontrolador smt32f4, pensando en un diseño propio más adelante entre el procesador de Texas intruments Omap3550 que se encargue de tener a la escucha el puerto serial y la implementación del filtro y por otro lado tareas menos pesadas para microcontrolador que maneje componentes eléctricos. Ya que esta versión es beta, los 2 componentes que necesitan del ADC (potencia y mezcla) para su funcionamiento se manejaran por medio de la stm32f4, ya que al utilizar los ADC con la Beaglebone, estos deben llevar un regulador de voltaje para que no quemen las salidas de estos perifericos que trabajan a 1.7 voltios. Este ejemplo del script de la tarjeta este hecho en lenguaje de programación C que es el que soporta el IDE de desarrollo, solamente se va a indicar el código que se ha trabajado directamente en el archivo main, ya que dicha tarjeta debe tener una configuración previa para poder utilizar los GPIOS y los ADCs, la configuración que se debe tener en cuenta para el script se muestra el siguiente ejemplo:

```
#define SW01_POS    0
#define SW02_POS    1
#define SW03_POS    2
#define SW04_POS    3
//se definen algunos nombres globales para el controlar los switches
uint16_t pins_val = 0;
//variable para comparar estados.
uint16_t pins_old = 0;
//variable para comparar estados.
Serial pc(USBTX, USBRX);
//se activa la comunicación serial de recepción y transmisión por el uart1
DigitalIn sw01(PA_14);
```

```
DigitalIn sw02(PA_15);
DigitalIn sw03(PB_7);
DigitalIn sw04(PC_13);
//a los nombres globales se les acciona pines de la tarjeta de desarrollo.
uint16_t updatePins(){
//funcion para actualizar estado de los pines
    uint16_t estado_pines = 0;
//variable para comparar estados.
    estado_pines += (sw01.read()) ? (1 << SW01_POS) : 0;
    estado_pines += (sw02.read()) ? (1 << SW02_POS) : 0;
    estado_pines += (sw03.read()) ? (1 << SW03_POS) : 0;
    estado_pines += (sw04.read()) ? (1 << SW04_POS) : 0;
//en la variable estado_pines se guarda el estado de cada switch, primero se lee lo que encuentre en el pin, si el valor es igual a 1, hay un corrimiento en el valor pero si es 0 el valor, el comparardor regresa también un 0, la idea es que la variable estado_pines represente en cada uno sus bits una entrada o switch.
    return estado_pines;
//retorna la variable estado pines
}

int main()
{
   millisStart();
   sw01.mode(OpenDrain);
   sw02.mode(OpenDrain);
   sw03.mode(OpenDrain);
   sw04.mode(OpenDrain);
//configuración de los pines para trabajar con interrupciones en cada uno de los perifericos.
  pc.baud (115200);
//velocidad de transimision 115200,
   pins_val = updatePins();
//asigna el en la variable la lectura actual de cada pin
   pins_old = pins_val;
// asgina en la variable la lectura anterior del grupo de pines
    while (1) {
// con este while, se asegura que siempre que haya un cambio de estado en los pines, este lo va registrat y ejectura alguna acción a ese cambio, por ejemplo envio de datos a través del serial con información para modificar valores en link2fs
            printf("E1%d\r\n",sw02.read());
            wait(0.01);
            printf("F02%d\r\n",sw03.read());
            wait(0.01);
            printf("C42%d\r\n",sw04.read());
            wait(0.01);
            printf("C43%d\r\n",sw05.read());
//tras cada cambio que lea, envia un carácter por el puerto serial para modificar el valor en cada pin
```

```
if(millis() - last_update > 100){
//se utiliza el contador interno del microcontrolador para generar esta acción cada cierto intervalo de tiempo, la idea es simular un delay pero sin parar todo el código en general, en el momento en que entra al condicional hace la lectura en cada ADC, y envia un valor formateado por el puerto serial acompañado de una clave que genera cambios en la mezcla de aire con gasolina o en el acelerador.

        printf ("C56%d\r\n", 100-(int)((adc1.read()-0.195)*125));
        wait(0.01);
        printf ("C58%d\r\n", 100-(int)((adc2.read()-0.18)*123));
        wait(0.01);
        last_update = millis();

//despues de ejecturar esta accio, la variable de tiempo se reinicia para dar una espera a que el tiempo vuelva a ser superior al del condicional para ejecturar de nuevo el condicional.
    }
```

C- Anexo de los esquemáticos diseñados para la elaboración del circuito.

Figura 3.15 esquemático de conexión del circuito superior.

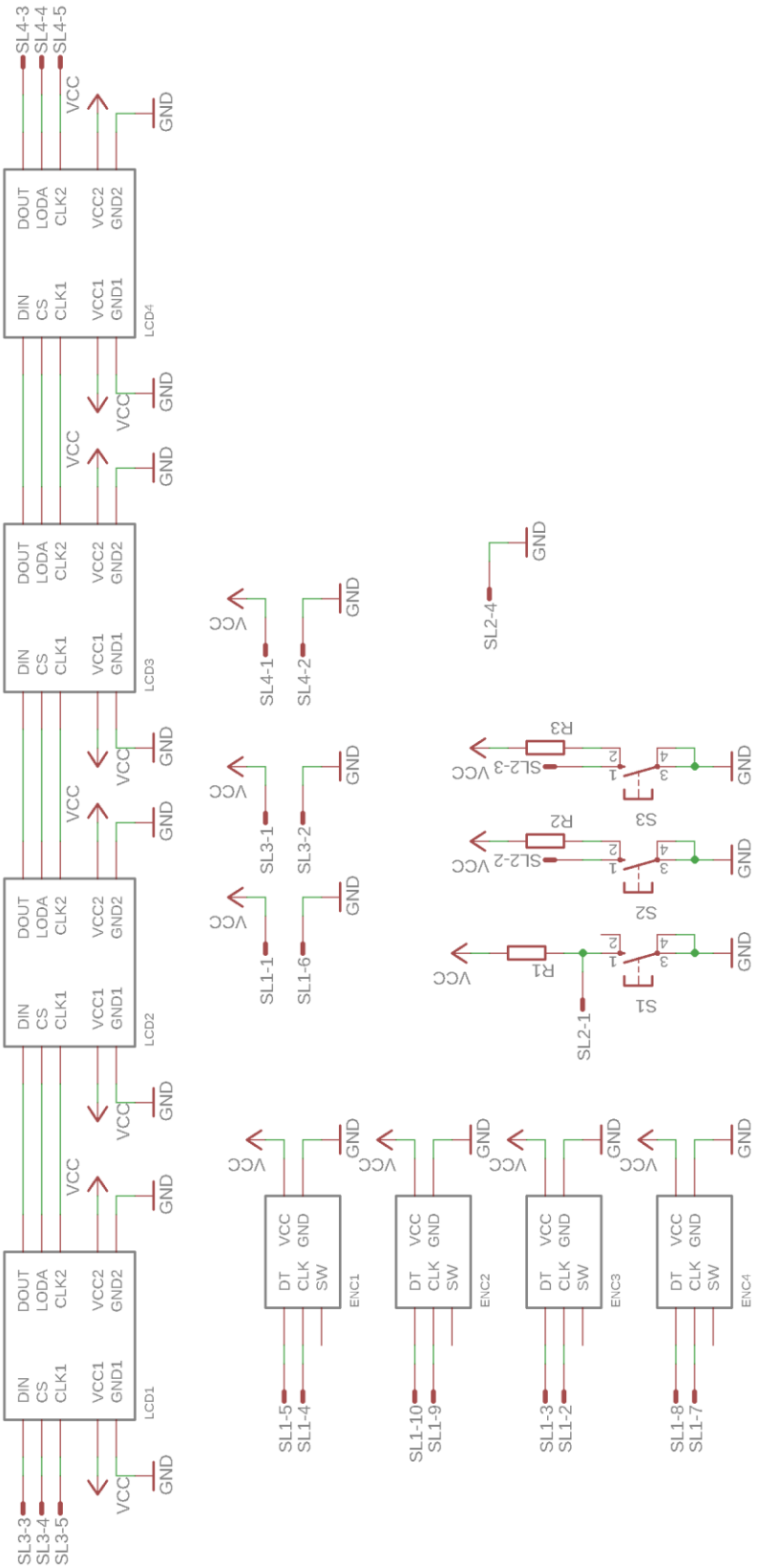

Figura 3.16 esquemático de conexión del circuito inferior.

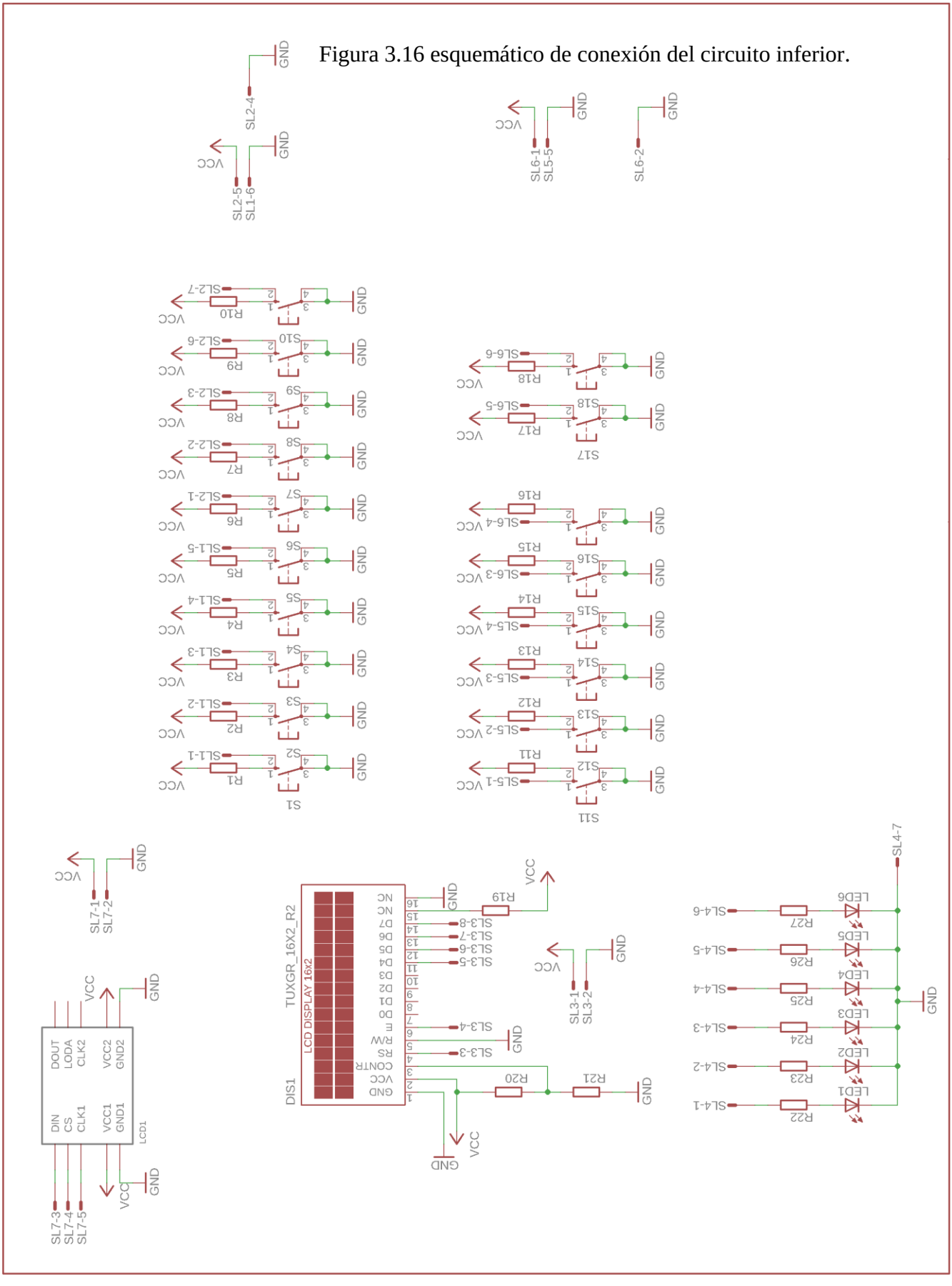

Figura 3.17 esquemático de posición del circuito superior.

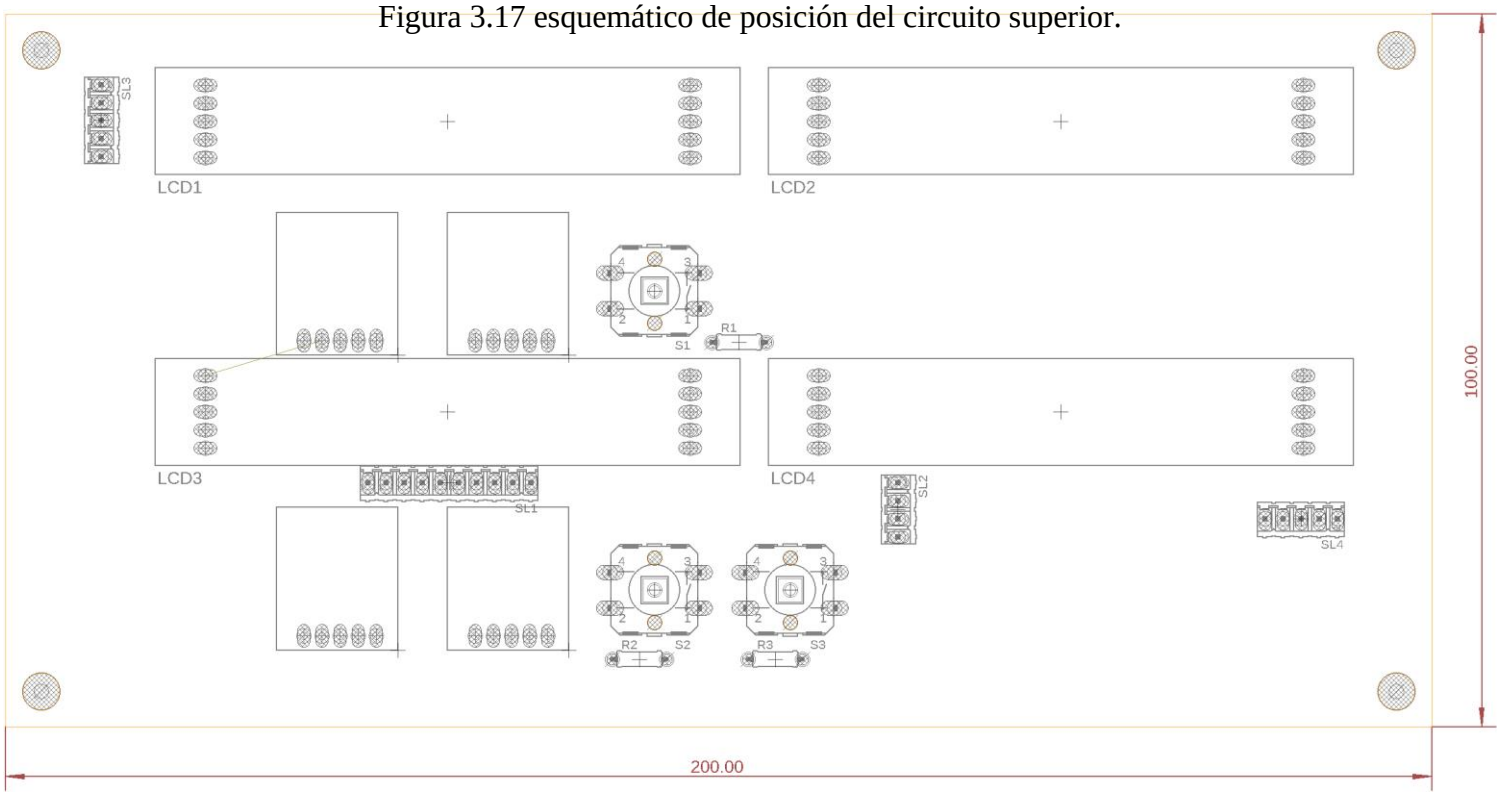

Figura 3.18 esquemático de posición del circuito inferior.

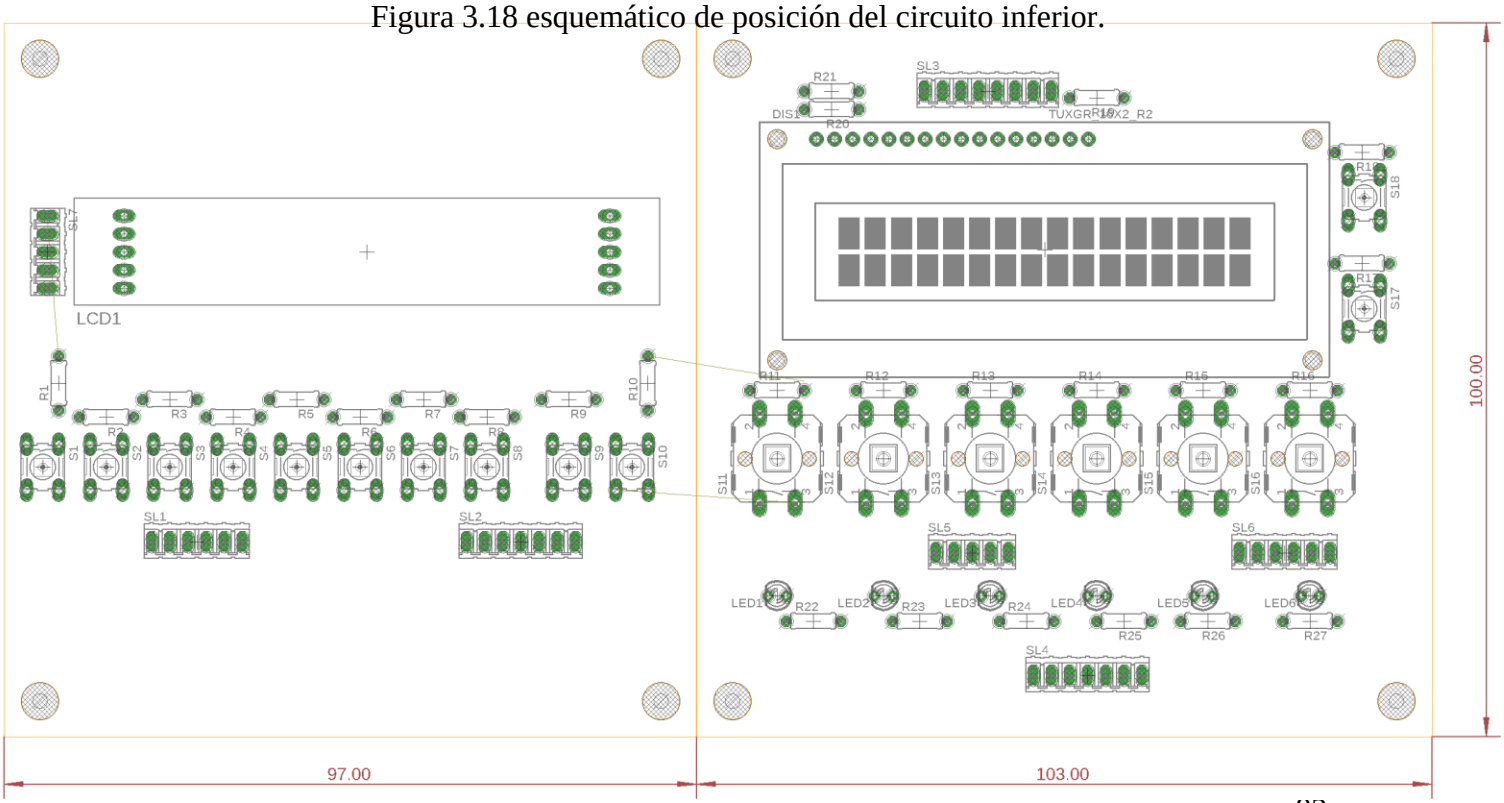

Figura 3.19 esquemático de pistas del circuito superior.

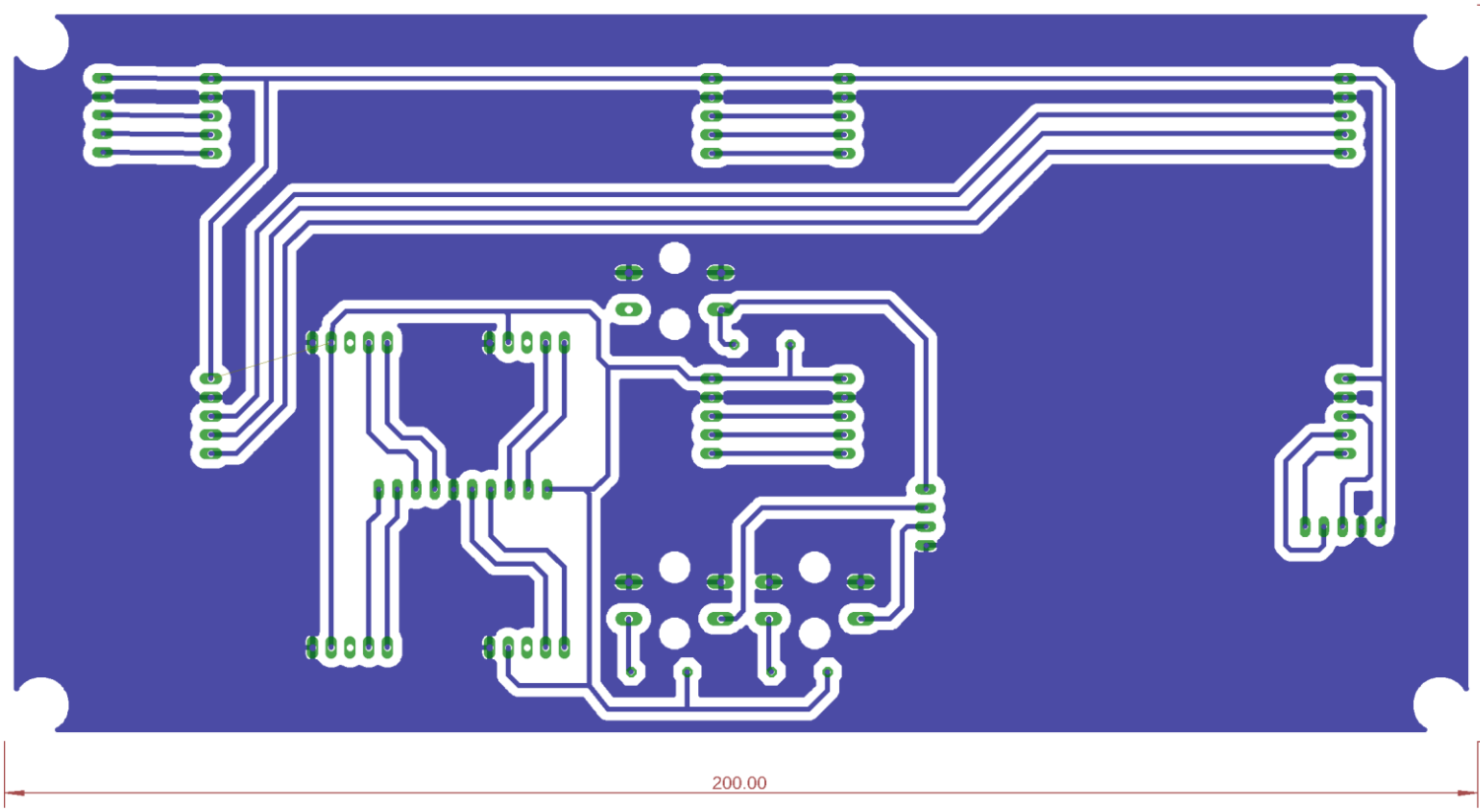

Figura 3.19 esquemático de pistas del circuito inferior.