RAE

1. **TIPO DE DOCUMENTO:** Trabajo de grado para optar por el título de INGENIERO AERONÁUTICO.
2. **TÍTULO:** DESARROLLO DE LA INTERCONEXIÓN BAJO PROTOCOLO CAN PARA EL BANCO DE AVIÓNICA DE LA UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ
3. **AUTORES:** Edgar Andrés Torres Sánchez y Luisa Fernanda Acero Martínez
4. **LUGAR:** Bogotá D.C
5. **FECHA:** 17 de junio del 2019
6. **PALABRAS CLAVE:** Aviónica, protocolo de comunicación, protocolo CAN, BUS CAN, microcontrolador, datos, banco de aviónica, transceptor.
7. **DESCRIPCIÓN DEL TRABAJO:** El presente proyecto busca desarrollar la interconexión usando el protocolo CAN como protocolo de comunicación en el banco de aviónica de la Universidad San Buenaventura Bogotá con el fin de unificar la comunicación del banco y así facilitar la recepción e interpretación de los datos recibidos por el usuario. Esto se realiza mediante el uso de un dispositivo que permite transformar el protocolo de salida de cada uno de los equipos al protocolo CAN, junto con esto, se plasma este documento con información física y operacional del dispositivo que también servirá como capítulo de comunicación del manual de operación del banco de aviónica.
8. **LÍNEA DE INVESTIGACIÓN:** Sistemas de aeronaves y teledetección
9. **METODOLOGIA:** Se usó metodología empírica y analítica, basándose en el diseño, programación y construcción de un dispositivo de enlace que usa un microcontrolador para realizar la conversión del protocolo de los dispositivos a protocolo CAN.
10. **CONCLUSIONES:** Es necesario realizar reparaciones y actualizaciones a los equipos del banco de aviónica puesto que la correcta operación del dispositivo se basa en la forma del funcionamiento de estos equipos. Los equipos que se añadirán posteriormente al banco de aviónica de la universidad deben contar con los requerimientos indicados en las recomendaciones debido que así se garantiza una integración exitosa en materia de comunicación con los demás equipos del banco de aviónica. Para el desarrollo de este tipo de dispositivos a nivel académico que manejan protocolo CAN, es altamente recomendable usar microcontroladores Microchip PIC, en especial de la familia 18 ya que su efectividad es bastante alta con relación al precio, mientras que para usos industriales se recomienda el uso de controladores CAN prefabricados y especializados ya que son especialmente diseñados para administrar el protocolo, aunque su precio es elevado. En el circuito generalmente se utiliza un optoacoplador el cual termina de refinar la señal del protocolo CAN, este no es necesario para el circuito y se comprobó que se puede obtener un resultado satisfactorio sin la implementación de este ya que los sistemas optoacoplados generalmente se utilizan para ambientes con mucha interferencia a causa del ruido. Para la programación del dispositivo se comprobó que son indispensables las interrupciones ya que permiten pasar de un módulo a otro dentro microcontrolador, las máscaras y filtros igualmente permiten que la información sea

discriminada y sin una correcta configuración de estos ítems, el microcontrolador no podrá operar bajo los parámetros requeridos. La PCB (Printed Circuit Board) es un elemento que permite reducir el tamaño, y cableado dentro del circuito, sin embargo, si no se pretende realizar una fabricación en masa, el costo es muy elevado, se recomienda que para dispositivos experimentales los montajes sean realizados en protoboard o en baquelita. Se demostró a través de la prueba de comunicación que el dispositivo es totalmente efectivo, cumple con los objetivos planteados al inicio del proyecto y puede ser implementado en el banco de aviónica, sin embargo, por la falta de acceso a los equipos del banco se sugiere realizar una actualización a dichos equipos con las recomendaciones aquí planteadas para que su enlace sea totalmente efectivo.

1. PRESENTACIÓN DEL PROYECTO

1.1. TÍTULO.

DESARROLLO DE LA INTERCONEXIÓN BAJO PROTOCOLO CAN PARA EL BANCO DE AVIÓNICA DE LA UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ

1.2. INTEGRANTES.

Luisa Fernanda Acero Martínez
Edgar Andrés Torres Sánchez

1.3. DIRECTOR.

Ing. Víctor Kenry Cruz Rodríguez

1.4. GRUPO DE INVESTIGACIÓN.

Aerotech

1.5. LÍNEA DE INVESTIGACIÓN.

Sistemas de aeronaves y teledetección

1.6. PROGRAMA.

Ingeniería Aeronáutica

1.7. DESCRIPTORES (PALABRAS CLAVES)

Aviónica, protocolo de comunicación, protocolo CAN, BUS CAN, microcontrolador, datos, banco de aviónica, transceptor.

DESARROLLO DE LA INTERCONEXIÓN BAJO PROTOCOLO CAN PARA EL BANCO DE AVIÓNICA DE LA UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ

Luisa Fernanda Acero Martínez

Edgar Andrés Torres Sánchez

UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ
FACULTAD DE INGENIERÍA
INGENIERÍA AERONÁUTICA
BOGOTÁ D.C.
2019

DESARROLLO DE LA INTERCONEXIÓN BAJO PROTOCOLO CAN PARA EL BANCO DE AVIÓNICA DE LA UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ

PROYECTO DE GRADO

Luisa Fernanda Acero Martínez

Edgar Andrés Torres Sánchez

DIRECTOR:

Ing. Víctor Kenry Cruz Rodríguez

UNIVERSIDAD DE SAN BUENAVENTURA BOGOTÁ
FACULTAD DE INGENIERÍA
INGENIERÍA AERONÁUTICA
BOGOTA D.C.
2019

Nota de aceptación

_____

_____

_____

_____

_____

Firma del presidente jurado

_____

Firma del jurado

_____

Firma del jurado

Bogotá D.C, junio 17 del 2019

DEDICATORIAS EDGAR

Primero que todo gradezco a Dios, él es el artífice y autor de toda esta historia.

A mi madre que, con su paciencia, sabiduría, apoyo y sobre todo con su amor, ha sido el pilar que me ha sostenido a lo largo de mi vida, ella es mi motivación para todo lo que hago, cada triunfo es en nombre de ella y por ella, sin ella nada de esto hubiera podido ser posible.

A mi padre que creyó en este proyecto de vida y me ha respaldado económicamente.

Y finalmente a Luisa que ha sido mi compañera, amiga y novia a largo de este trayecto, ella ha sido una luz en mi camino y una sombra que ha acompañado mis pasos, juntos iniciamos y atravesamos esta senda y juntos la culminamos.

DEDICATORIAS LUISA

En principio quisiera dedicar y agradecer el desarrollo y logro de este proyecto a Dios porque sin su ayuda nada de esto hubiera sido posible; a mi papá por el apoyo económico, a mi mamá, mi hermana y mi abuelita porque siempre creyeron en mí, me apoyaron en los momentos difíciles y han sido mi mayor motivación para esforzarme y lograr cada una de las cosas que me he propuesto a lo largo de la carrera y a Edgar porque siempre fue mi compañero, mi amigo, mi amor y codo a codo fuimos un apoyo para el otro hasta que el día de hoy podemos decir ¡lo logramos!

AGRADECIMIENTOS

Agradecemos a todas las personas que hicieron posible este proyecto, en especial al ingeniero Víctor Cruz por su acompañamiento y apoyo, a los ingenieros Wilson Pinzón, Fabio Merchán y Myriam Tarazona por sus orientaciones, al ingeniero José David Alvarado por su ayuda e información y a Jeffer Sánchez por sus múltiples colaboraciones.

LISTA DE CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ECUACIONES

TABLA DE SIGLAS Y SIMBOLOS

Símbolo	Significado
CAN	Controller Area Network
Tx	Línea de transmisión
Rx	Línea de recepción
CPU	Central Processing Unit
ROM	Read Only Memory
CAC	Convertidor de señales analógicas a digitales
CDA	Convertidor de señales digitales en analógicas
CANH	CAN High
CANL	CAN Low
UTP	Unshilded twister pair
Ω	Ohms
CiA	CAN in automation
ISO	International Organization for Standardization
GND	Ground-Tierra
t_q	Time Quanta
t_{bit}	Time bit
BRP	Baud rate prescalator
MSB	Most significant bit
LSB	Low significant bit
CRC	Código cíclico de redundancia
PWM	Pulse width modulation
ECAN	Enhanced CAN
IDE	Integrrated development enviroment

1. ANTECEDENTES

1.1. Históricos

En los últimos años se ha presentado un salto importante del paso de señales analógicas a señales digitales, este desarrollo comenzó a causa de la necesidad de los creadores de dispositivos de funcionamiento analógico de tener la capacidad de poder emular el comportamiento de los dispositivos sin efectuar un costoso prototipo, el medio por el cual se llevaban a cabo estas simulaciones era la computadora y así fue como se empezó el adelanto de procesamiento de ondas digitales. Los pioneros de este desarrollo empezaron a trabajar en esto alrededor de principios de la década de los 50 con la invención del transistor que se muestra en la *figura 1* y a lo largo de los años 60 cuando los militares tenían sistemas informáticos, pero no se imaginaban que esto se expandiría hacia un el campo mayor de la electrónica digital en la década de los 80 y consiguientes.

Figura 1.Modelo del primer transistor. Tomado de: https://www.researchgate.net/figure/Figura-10-Primer-transistor-6_fig2_271053621

.

En 1969 aparece el ARPANET, siendo la primera vez que el público conociera los conceptos que llevarían a la creación del internet, estas redes de paquete conmutado se desarrollarían utilizando una gran variedad de protocolos. En 1983 la empresa Robert Bosch GmbH comenzaría el desarrollo del protocolo CAN y este sería lanzado en 1986 y para 1987 aparecerían los primeros controladores CAN como el que se encuentra en la *figura 2*. (“Historia del transistor,” 2007)

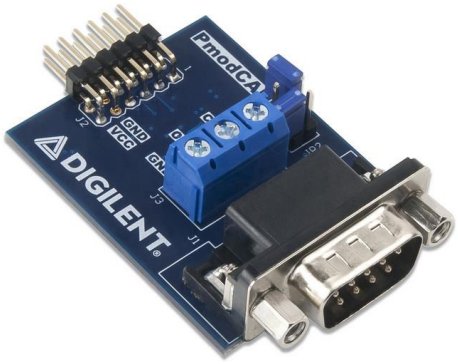


Figura 2. Controlador CAN. Tomado de: https://store.digilentinc.com/pmod-can-can-2-0b-controller-with-integrated-transceiver/

Debido al éxito que tuvo este protocolo dentro de la industria automotriz, se empezó a implementar en otras industrias y por su versatilidad Airbus lo implementó en el superjumbo A380 (*figura 3)*. (Rao, 2009)

Figura 3. Superjumbo A380. Tomado de: https://www.super-hobby.es/products/Airbus-A380-First-Livery.html

1.2. Internacionales

A nivel mundial se han llevado a cabo varias investigaciones acerca del protocolo CAN siendo reconocidas universidades como: University of Porto y University of Aveiro; siendo estas las principales fuentes de estos estudios ya que la mayoría de las investigaciones son privadas o gubernamentales sin acceso al público. Se destaca el documento: The FTT-CAN Protocol: Why and How (Almeida, Pedreiras, & Fonseca, 2002). En donde se plantea la necesidad de flexibilidad en los sistemas de bus de campo dentro de las industrias actuales en todos los niveles de los sistemas que incluyen el nivel de campo dentro de los procesos y los niveles de control de células, esto con el fin de reducir costos en la configuración como se ve en la *figura 4*, cambios de configuración y el mantenimiento de los sistemas por lo que justifican la creación de un nuevo protocolo de comunicación denominado:

Flexible Time-Triggered communication on Controller Area Network.

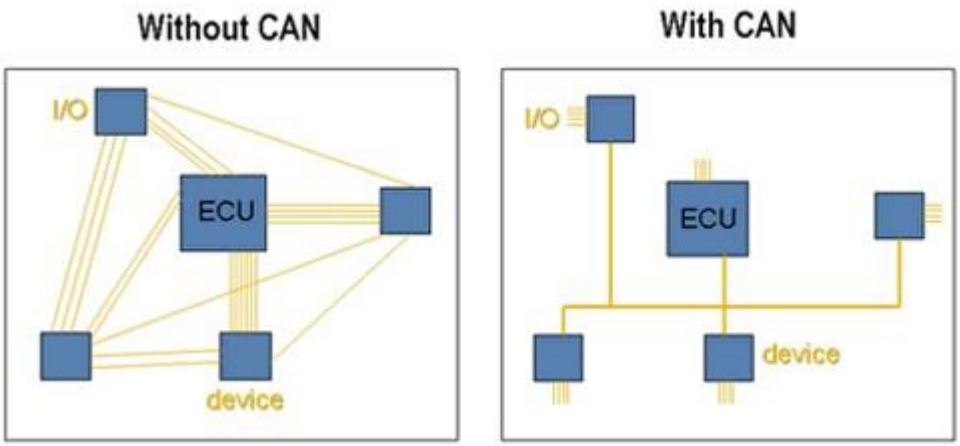


Figura 4. Beneficios del protocolo CAN. Tomado de: http://www.ni.com/es-co/innovations/white-papers/06/controller-area-network--can--overview.html

1.3. Regionales

A nivel latinoamericano una de las universidades más reconocidas en este tipo de investigaciones es la universidad Nacional de la Plata en Argentina, la cual realizó un proyecto denominado Análisis y simulación de un protocolo de comunicaciones para una red de sensores (Diego Encinas, Frati, & Naiouf, 2011). En el cual se realiza un estudio y clasificación de los protocolos para posteriormente realizar una descripción del protocolo CAN definiendo los principios de modelamiento y simulación enfocados a los sistemas aeroespaciales. Finalmente llevan a cabo los experimentos en dos entornos: hardware y simulación.

También el proyecto denominado Aplicación en dispositivos de vuelo y el protocolo de comunicaciones CAN aplicado a satélites y vehículos lanzadores (D Encinas & Meilan, 2009); en donde se presenta una utilidad del protocolo CAN con el fin de ser puesto en práctica en un dispositivo aeroespacial estudiando sus atributos dentro de una conexión satelital y cohetes de lanzamiento de un modelo de prueba usando una red de microcontroladores conectados al bus CAN por medio de controladores y transceptores. Por otro lado, a nivel nacional se destacan la Universidad Nacional y la Universidad de Medellín, cuyas investigaciones a resaltar son un proyecto que consistía en implementar el protocolo CAN a un aeromodelo para así permitir comunicar lecturas de altura, presión, velocidad, etc. En la Universidad Nacional se realizó el análisis de una subred para comunicaciones aeronáuticas en las terminales aéreas de las ciudades más importantes de Colombia (Fernando & Ortiz, 2016); en donde se presentan las conclusiones de una representación simulada para emular una subred de estaciones fijas para la telecomunicación aeronáutica basadas en las ideas de operación del organismo OACI y las demandas de operación de prestador de servicios en la aeronavegación Colombiana utilizando el instrumento de emulación

OPNET y finalmente en la Universidad de San Buenaventura en el 2008 se realizó un proyecto en el cual se diseñó un banco de aviónica (Cely, Allain; Ojeda, Martiza; Jiménez, 2008). En el cual se diseñó un laboratorio didáctico de aviónica y comunicación con el fin de mejorar la forma de aprendizaje sobre los instrumentos de radio y comunicación.

Figura 5. Proyectos Universidad Nacional de la Plata y Universidad de San Buenaventura. Tomado de: Cely, Allain; Ojeda, Martiza; Jiménez, J. (2008). DISEÑO DE UN LABORATORIO DIDACTICO DE AVIONICA PARA LA UNIVERSIDAD SAN BUENAVENTURA; Encinas, D., & Meilan, P. (2009). Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores. XV Congreso Argentino

2. DESCRIPCIÓN Y FORMULACIÓN DE LA PREGUNTA O PROBLEMA DE INVESTIGACIÓN

Este proyecto se llevará a cabo en el banco de aviónica de la Universidad de San Buenaventura Bogotá durante el primer semestre del año 2019, con la necesidad de homogeneizar y unificar el protocolo de comunicación de los equipos del banco de aviónica de la universidad para una identificación individual de los mismos dentro de un controlador maestro, resaltando que el protocolo a implementar tenga uso aeronáutico y sea de uso libre se plantea la siguiente pregunta de investigación:

¿Cómo unificar el protocolo de comunicación en el banco de aviónica de la Universidad de San Buenaventura Bogotá?

3. OBJETIVOS

3.1. OBJETIVO GENERAL:

Desarrollar la interconexión usando el protocolo CAN como forma de comunicación entre los equipos del banco de aviónica de la Universidad de San Buenaventura Bogotá.

3.2. OBJETIVOS ESPECÍFICO

3.2.1. Evaluar de forma operativa y funcional los equipos del banco de aviónica de la Universidad de San Buenaventura Bogotá.

3.2.2. Establecer un dispositivo de enlace entre los equipos y el bus CAN, cuya función sea transformar los datos desde el protocolo que posea el equipo al protocolo CAN.

3.2.3. Crear un capítulo de manual de operación en donde se encuentre las características físicas y de funcionamiento del dispositivo.

3.2.4. Plantear una estructura de comunicación para los equipos de aviónica que permita bajo actualización posterior la comunicación a través del protocolo CAN.

4. ALCANCES Y LIMITACIONES

4.1. ALCANCES

4.1.1. El presente proyecto busca unificar las comunicaciones dentro del banco de aviónica de forma tal que permita una mejor capacidad de instalación e integración de los equipos.

4.1.2. A partir de la implementación del protocolo se elaborará este documento con información acerca del funcionamiento del dispositivo y sus características de comunicación, este documento será el precedente como capítulo de comunicación del manual de operación del banco de aviónica de la Universidad de San de Buenaventura Bogotá.

4.1.3. Construir físicamente el dispositivo de interconexión bajo el protocolo CAN.

4.2. LIMITACIONES

4.2.1. La implementación está sujeta al correcto funcionamiento de los dispositivos en el banco de Aviónica, no se harán reparaciones ni operaciones invasivas en los dispositivos.

4.2.2. Los dispositivos que se añadirán posteriormente al banco de aviónica tendrán que acoplarse siguiendo información de este proyecto que es el respectivo documento de información para el manual de operación del banco de aviónica.

4.2.3. Se plantea un presupuesto hasta de un millón de pesos ($1´000.000) para llevar acabo el desarrollo y prueba del protocolo CAN.

4.2.4. La prueba de comunicación se llevará a cabo mediante un dispositivo de recepción de datos CAN adquirido o mediante un software de simulación de recepción de datos CAN, esto sujeto a la pertinencia y conveniencia económica y temporal del proyecto.

4.2.5. La prueba de comunicación se llevará a cabo mediante un dispositivo de recepción de datos CAN adquirido o mediante un software de simulación de recepción de datos CAN, esto sujeto a la pertinencia y conveniencia económica y temporal del proyecto.

4.2.6. El documento de información para capítulo de manual de operación del banco de aviónica es el mismo documento presentado para proyecto de grado y solo

contendrá información física y operacional del dispositivo de empalme y como este se integra a los equipos del banco de aviónica.

4.2.7. Se limita la evaluación operativa y funcional a identificar el protocolo que usan los dispositivos, la estructura del mensaje y el correcto encendido y apagado de la unidad.

4.2.8. Solo se transformarán datos desde el protocolo serial a protocolo CAN.

5. MARCO TEÓRICO

5.1. Protocolos de comunicación

Como se muestra en la *figura 6*, el objetivo principal de la comunicación es poder llevar información de un punto a otro y que esta información llegue sin errores, para esto se utilizan múltiples canales que crean un vínculo entre los dos puntos. Los puntos son los espacios donde se encuentran los dispositivos transmisores y receptores, así como los equipos que se encargan de la codificación y decodificación de la información.

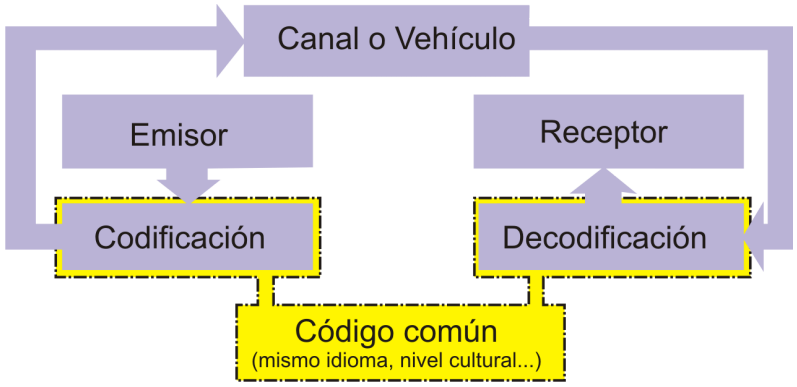


Figura 6. Comunicación. Tomado de: http://introducioncomunicacion.blogspot.com/2015/11/la-comunicacion-y-codificacion-y.html

Dentro de la comunicación hay muchos dispositivos y factores que intervienen para que esta se lleve a cabo, uno de los componentes más importantes dentro del sistema que permite que la comunicación se lleve a cabo es lo que se conoce como el protocolo de comunicación.

Un protocolo de comunicación es un conjunto de normas y reglas para la transmisión de datos entre dos o más puntos y estas normas gobiernan todo el intercambio de información para que esto se haga de forma ordenada y sin errores.

Un protocolo se compone básicamente de los siguientes elementos: conjunto de caracteres, normas para la secuencia y sincronización de la información, los niveles voltaje y los procedimientos para la detección y corrección de errores. Para el conjunto de caracteres se tienen de dos tipos, caracteres de significado o imprimibles y los caracteres de control y generalmente cada protocolo tiene predefinido la equivalencia de caracteres y su codificación binaria que se plasma en el código. Las normas de sincronización permiten que la información se transmita de manera ordenada y a tiempo haciendo que el transmisor y receptor tengan una operación simultanea y uniforme. Los procedimientos de detección y corrección de errores permiten identificar cuando la información no se está transmitiendo de forma satisfactoria por factores sin control en alguno de los actores de la comunicación,

generando una alerta y si es el caso permite que el error sea corregido y la información recuperada.

Cabe resaltar que para que haya una comunicación entre dos puntos, los dos puntos tienen que usar el mismo protocolo de comunicación ya que se puede hacer la analogía de que el protocolo es el idioma con el cual se van a comunicar por lo que se podrá decir que ambos puntos "tienen que hablar el mismo idioma".

Existen múltiples protocolos, pero se abordarán dos principalmente: el protocolo CAN y el protocolo RS-232; a continuación, se hará profundización en sus características principales y modos de operación.

5.2. Protocolo CAN

El protocolo CAN (Controller Area Network) nació en 1985 a manos de la empresa Bosch, fue desarrollado para las redes de comunicación de dispositivos de vehículos automotrices ya que antes de la llegada de este protocolo de comunicación los industriales del parque automotor enlazaban los dispositivos electrónicos usando redes de cables punto a punto como en la *figura 7*, pero a medida que los industriales empezaron a usar muchos más dispositivos electrónicos, los cableados aumentaban en gran medida el peso y el costo de los automotores así que al cambiar este sistema de cables por sistemas de comunicación con protocolos como se muestra en la *figura 8*, se logró disminuir los costos de fabricación, la complejidad de la conexión y el peso, debido a esto la industria automotriz acogió rápidamente el protocolo CAN haciendo que este tuviera la categoría de estándar mundial como ISO 11898 y se mudara a otras industrias como la industria textil y la industria aeronáutica llevando a que a partir de 1994 se desarrollaran protocolos de más alto nivel con base en el protocolo CAN como lo son el CANopen y el ARINC 825.

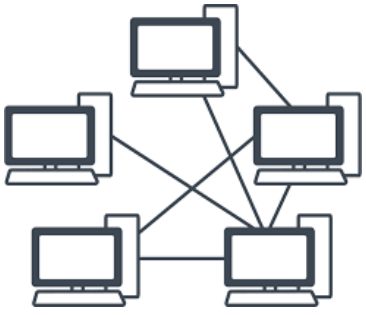


Figura 7. Conexión punto a punto. Tomado de: https://www.lucidchart.com/pages/es/que-es-un-diagrama-de-red

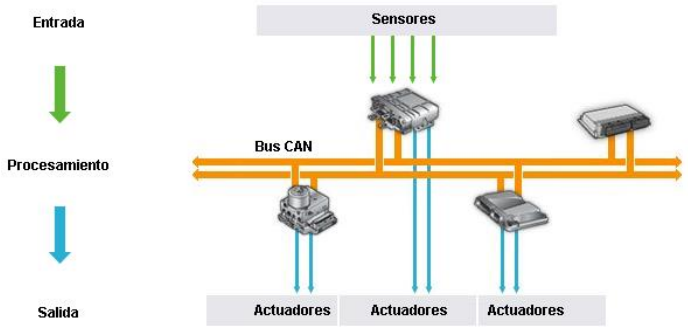


Figura 8. Ejemplo de bus CAN. Tomado de: http://jairodaniel25.blogspot.com/2015/09/can-bus.html

El protocolo CAN provee al usuario la ventaja de tener una red de comunicación altamente robusta y económica, además, permite que varios dispositivos de la red puedan comunicarse entre sí enviando y recibiendo información por igual evitando tener múltiples cableados para cada requerimiento sin mencionar el inconveniente de las entradas y salidas tanto analógicas como digitales por la diferenciación de los dispositivos.

El protocolo CAN está considerado como una red uno a uno lo que quiere decir es que se puede o no tener un sistema maestro-esclavo ya que cualquiera de los nodos dentro de la red tiene acceso a la lectura o escritura de información sobre el bus. Cuando uno de los dispositivos dentro de la red quiere transmitir datos, primero se cerciora de que el bus este desocupado, y si es así procede a escribir una trama de datos sobre la red, estas tramas no tienen una dirección en específico desde el nodo que transmite hacia cualquiera de los posibles receptores, sencillamente la trama cuenta con un espacio de identificación conformada por máscaras y filtros únicos que lo que hacen es que todos reciban la trama pero dependiendo del identificador el nodo rechace o acepte esta información.

Figura 9. Mercedes-Benz clase e 1992, primer automovil en implementar el protocolo CAN. Tomado de: http://www.cars-directory.net/gallery/mercedes/e-class/1992/mercedes_e-class_a1212500517b1741196_p.html

5.3. RS232

El protocolo RS-232 es un protocolo de estándar internacional que gobierna los factores de un modo de comunicación serial. A través de este protocolo se crean los estándares en cuanto a la velocidad de transmisión de datos, la forma de controlar esta transmisión, niveles de voltaje usados, los cables para la transmisión de este protocolo, distancia entre los equipos, formas de conectar, conectores y demás.

Una de las características de la comunicación serial es que aparte de la línea de transmisión (Tx) y la línea de recepción (Rx) se tiene otra línea denominada de control de flujo (Handshake) y su utilización es optativa según los requerimientos y dispositivos que se quieren conectar.

El protocolo RS-232 transmite información de forma asincrónica y es necesario añadir unas señales definidas dentro de las reglas del protocolo. Los niveles de tensión usados están dentro del rango de -15 hasta los 15 voltios.

Normalmente el protocolo RS-232 emplea conectores DB9 de 9 pines para la comunicación, en la siguiente tabla se muestra la designación de cada señal en los pines del conector

Tabla 1. Designación de señales y pines RS-232

Señal	# PIN
Ring indicator	9
Clear to send	8
Request to send	7
Data set ready	6
Signal ground	5
Data terminal ready	4
Transmit data	3
Receive data	2
Carrier detect	1

A continuación, se da una breve descripción de las principales señales que actúan en este protocolo:

5.3.1. Request To Send

Esta es la conexión de la señal cumple la función de indicar que se desea enviar información, si el otro dispositivo receptor está listo para empezar a recibir datos se inicia la transmisión de un dispositivo a otro.

5.3.2. Clear To Send

Esta es la conexión de la señal que indica que el dispositivo receptor está listo para recibir los datos.

5.3.3. Data Terminal Ready

Esta es la conexión de la señal que indica que el dispositivo emisor terminó de enviar datos y está listo para empezar a recibir los datos.

5.3.4. Data Set Ready

Esta es la conexión de la señal que es la respuesta a la señal Data Terminal Request y le indica al dispositivo receptor que el emisor ya está listo para recibir información

5.3.5. Receive Signal Line Detect

Esta es la conexión de la señal que le indica al dispositivo emisor por parte del dispositivo receptor que se ha creado una conexión con otro dispositivo receptor adicional

5.3.6. Transmit Data

Esta es la conexión de la señal que transporta la información y se envía de a un bit cada vez

5.3.7. Receive Data

Esta es la conexión de la señal que recibe la señal de a un bit cada vez

5.4. MICROCONTROLADORES

Como se muestra en la *figura 10*, los microcontroladores son dispositivos que contienen un circuito integrando dentro de sí, estos circuitos están constituidos por una Central Processing Unit (CPU), memorias Read Only Memory (ROM), puertos de entrada y de salida además contiene unidades denominadas periféricos. Todos estos componentes están conectados entre sí dentro del microcontrolador, además, el microcontrolador posee una memoria en la cual se incorpora una programación para que este funcione de una forma específica y realice funciones determinadas.

Las entradas del microcontrolador son usadas para acoplar sensores u otras fuentes de información que requiere ser procesada y así mismo las salidas son usadas para realizar el control o la comunicación con cualquier tipo de dispositivo.

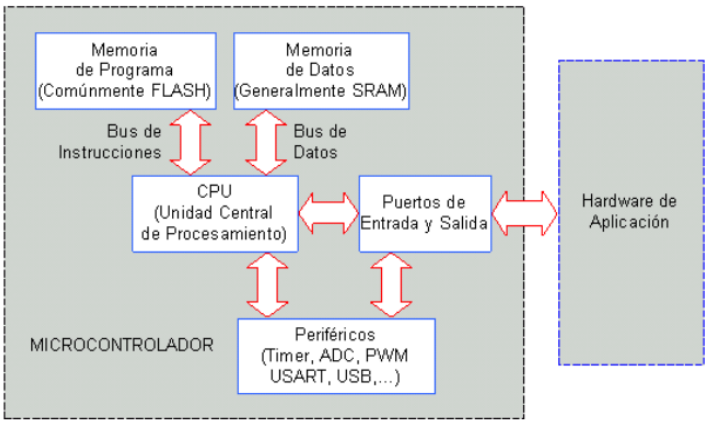


Figura 10. Arquitectura de un microcontrolador típico. Tomado de: http://www.exa.unicen.edu.ar/catedras/tmicrocon/Material/1_introduccion_a_los_ucontroladores.pdf

La función de un microcontrolador es interpretar y realizar las ordenes propiciadas por el usuario y esto lleva a que la programación de un microcontrolador reduzca la complejidad de los circuitos electrónicos. Dentro de un microcontrolador se encuentran múltiples instrumentos que permiten que este funcione de la mejor manera, estos instrumentos son los siguientes:

5.4.1. Puertas de comunicación

Esta herramienta se utiliza para poder interconectar el microcontrolador con cualquier tipo de elemento externo, como por ejemplo buses de datos

5.4.2. Puertas de entradas y salidas digitales

Estas líneas que posee el microcontrolador pueden ser configuradas ya sea bien como entrada o salida de acuerdo a los registros propiciados por el fabricante.

5.4.3. Comparador analógico

A veces los microcontroladores tienen amplificadores operaciones que sirven como comparadores entre dos señales

5.4.4. CAC

Este es un convertidor que se encarga de transformar una señal analógica en una señal digital y le permite al microcontrolador trabajar con señales de tipo analógico.

5.4.5. CDA

Este es un convertidor que se encarga de transformar una señal digital en una señal analógica y le permite al microcontrolador trabajar con señales de tipo digital.

5.4.6. Sleep

Los microcontroladores utilizan una función denominada Sleep, que le permite ahorrar energía. Esto lo logra desactivando elementos internos como el reloj, este modo puede ser desactivado cuando un evento esperado ocurra lo que lleva a que el microcontrolador regrese a su estado normal.

Los microcontroladores usan dos arquitecturas principalmente para estructurar el funcionamiento, las cuales son la arquitectura tipo Harvard y la arquitectura tipo Von Neumann

La arquitectura Harvard (*figura 11)* se caracteriza por tener una memoria que solo es exclusiva para las instrucciones y una memoria aparte que es exclusiva para los datos, usando 2 Buses de datos diferentes permitiendo que el microcontrolador use las dos memorias al mismo tiempo ganando velocidad en el procesamiento de la información.

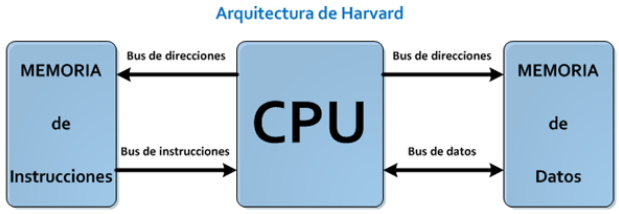


Figura 11. Esquema de arquitectura Harvard. Tomado de: https://www.electrontools.com/Home/WP/2018/04/15/diferencias-entre-los-modelos-de-von-neumann-y-harvard/

La arquitectura Von Neuman (*figura 12)* se caracteriza por circular las instrucciones y los datos a través un único bus común para ambos, su ventaja principal es la disminución de líneas sacrificando la velocidad de procesamiento.

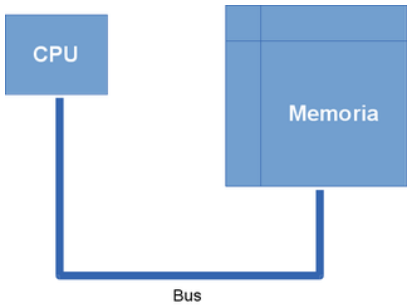
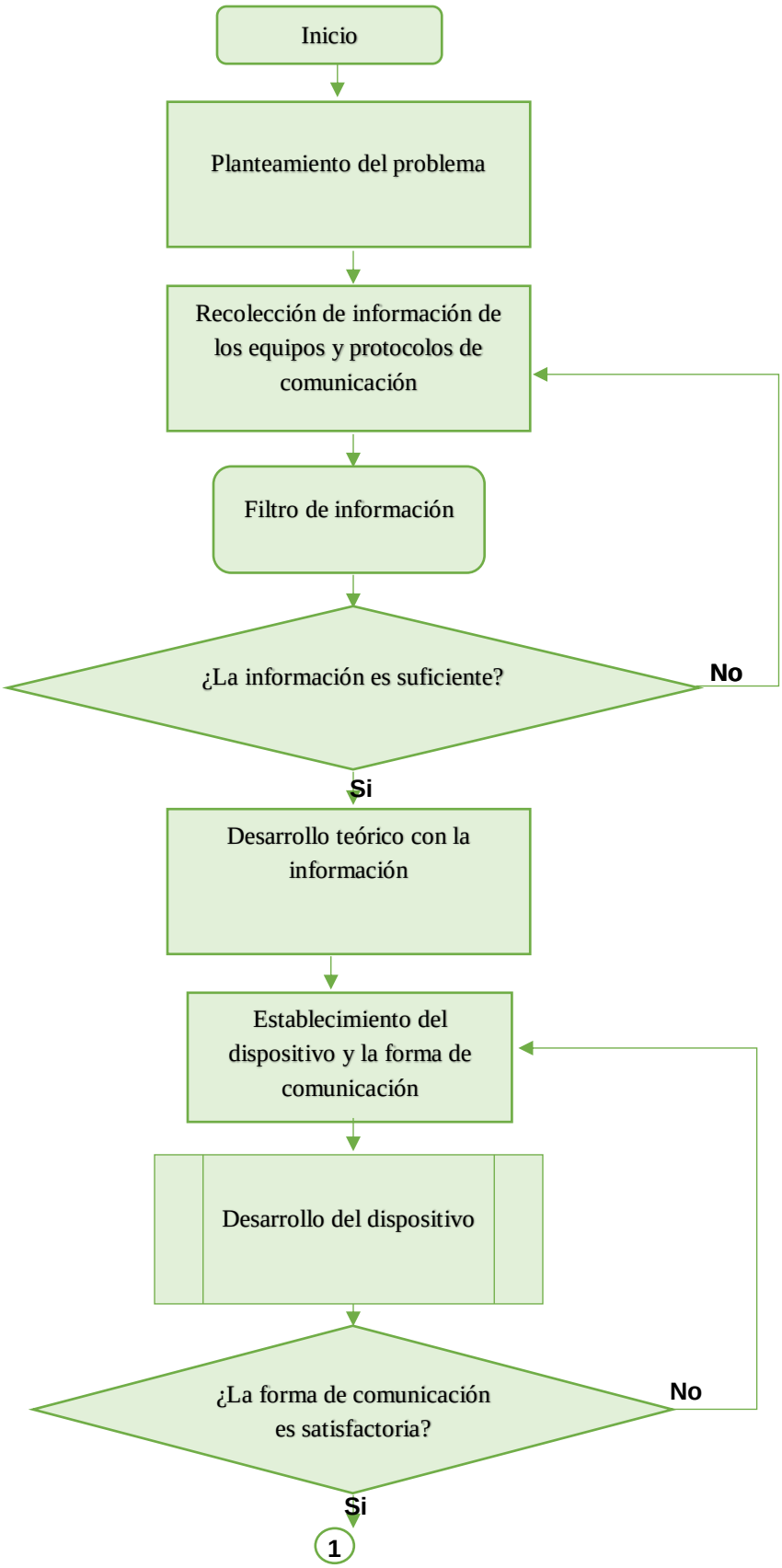
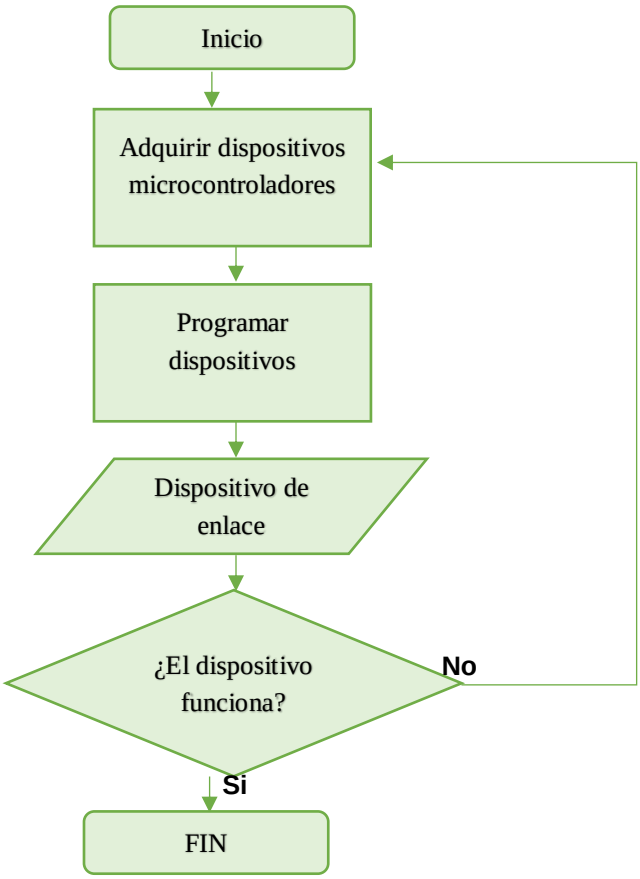


Figura 12. Esquema de arquitectura Von Neuman. Tomado de: https://frikosfera.wordpress.com/2015/02/27/que-es-la-arquitectura-von-neumann/

6. METODOLOGIA



6.1. Sub-Rutina de desarrollo del dispositivo



Este proceso fue indicado como una subrutina puesto que es el eje principal del proyecto e implica el desarrollo y pruebas de este, proceso que se debe repetir hasta que el dispositivo funcione de forma correcta.

7. DESARROLLO DE INGENIERÍA

7.1. Evaluación de equipos

Los equipos del banco de aviónica se componen de un AIR DATA COMPUTER cuya función dentro del banco de aviónica es administrar los datos obtenidos de los sensores de aire y enviarlos computados a cada uno de los equipos indicadores que se encargan de suministrar esta información por medio de una interfaz al usuario, una EADI cuya función es integrar múltiples dispositivos de navegación en donde se indica la posición de vuelo junto a otro datos de información relevante como la altitud, referencia de vuelo, posición de cabeceo, etc y finalmente un Vertical Speed Indicator o variómetro cuya función es indicarle al usuario la velocidad de ascenso o descenso de la aeronave (VSI). Adicionalmente se determinó por medio del tutor de tesis que posteriormente se iban añadir más equipos a corto y a mediano plazo.

Se pretendía realizar una manipulación y prueba funcional de los equipos del banco de aviónica, pero no fue posible ya que no se encontraban disponibles. Esto debido a la incompatibilidad de uno de los equipos con la estructura de envío de información y al daño en la parte electrónica de otro de los dispositivos, esta información fue proporcionada por el tutor de tesis , por lo que se optó por realizar una recolección de información de los proyectos con los que se realizaron estos equipos sin embargo estos documentos de proyecto de grado con los cuales se realizaron estos dispositivos son de carácter privado por lo que no se pudo acceder a ellos.

Finalmente se optó por acceder a la información acerca de la forma de comunicación a través del tutor de tesis ya que el asesoró los grupos que desarrollaron estos equipos.

Se determinó por medio del tutor de tesis que estos equipos tenían como como salida de datos el protocolo serial RS-232, adicionalmente se determinó que estos dispositivos enviaban la información sin identificadores por lo que es necesario realizar modificaciones con las recomendaciones planteadas en este documento.

El tutor informó adicionalmente, que estos equipos se encuentran inoperativos por lo que se recomienda que inicialmente se les realice una limpieza y mantenimiento, así como reparaciones a los equipos que así lo requieren.

7.2. Desarrollo teórico

7.2.1. Niveles de tensión en el bus CAN

Para la transmisión y recepción de información es necesario codificar la información a través de niveles de tensión definidos, para el caso del protocolo CAN se usan los criterios de bit recesivo y dominante, esta es la definición para el caso del protocolo CAN:

Recesivo: el recesivo es un bit cuyo valor representa un “1” lógico

Dominante: el dominante es un bit cuyo valor representa un “0” lógico y supone su representativa dominante ya que es capaz de suprimir los “1”

En materia física el bus usa una línea eléctrica para cada uno de los bits y se llaman CAN HIGH y CAN LOW (*figura 13)*, la señal de lectura de bit se realiza restando el CAN HIGH y el CAN LOW y esto hace que este protocolo sea blindado a la interferencia electromagnética y elimina la tierra como una necesidad puesto que el CAN HIGH es referencia del CAN LOW.

Existen usos donde es necesaria una velocidad de transmisión alta que esta del rango de 125 kbit/s a 1 Mbits/s y se le llama high-speed CAN y los usos donde es necesaria una transmisión menor a 125 kbit/s se le denomina FT LS CAN physical layer.

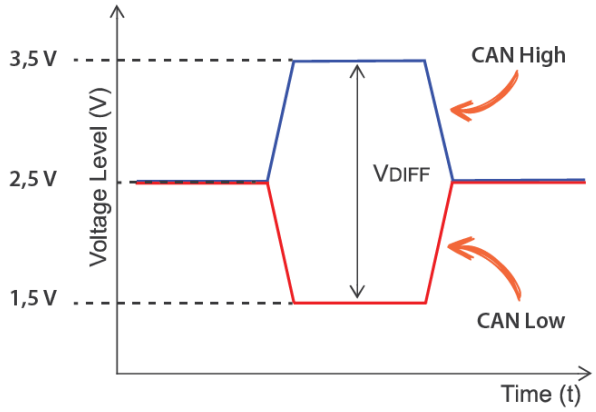


Figura 13. Niveles de tensión CAN. Tomado de: https://www.kmpdrivetrain.com/paddleshift/practical-tips-can-bus/

7.2.2. Bus físico

Para la transmisión de protocolo CAN generalmente se utiliza para el bus un cable trenzado denominado unshilded twister pair (UTP). Aunque se pueden usar muchos cables diferentes en la prueba de comunicación de este proyecto se utilizó uno trenzado ya que por normatividad estándar para el protocolo CAN se utiliza este tipo de cable, adicionalmente se usaron unas resistencias 120Ω en los extremos, esto se recomienda para evitar los reflejos dentro del bus ya que es un problema común dentro de las redes CAN, el valor resistencias implementadas en este proyecto se eligió debido a que para este proyecto se va usar una velocidad de transmisión baja porque para uso de high-speed CAN es necesario valores de resistencia desde 150 a 300 Ω.

Figura 14. Cable UTP. Tomado de: https://www.ecobadajoz.es/cables-ethernet/bobina-cable-utp-cat-6-outdoor-23awg-300-metros-mod-mc846n300.html

7.2.3. Conectores

El conector recomendado para el uso de este protocolo según la CiA (CAN in automation) junto a los estándares ISO especificados es un conector sub D 9 y para este proyecto se utilizó un conector de este tipo ya que en la mayoría de industrias se siguen las recomendaciones y estandarización realizadas por la CiA y la ISO para la implementación del protocolo dentro de una red.

Según el diseño del conector recomendado por la CiA los pines tienen la siguiente función

Tabla 2. Pines del conector sub D 9

1	-	Reservado
2	Can_L	Línea del bus dominante (low)
3	CAN_GND	Tierra del CAN
4	-	Reservado
5	CAN_SHLD	Escudo CAN (opcional)
6	(GND)	Tierra del CAN (opcional)
7	CAN_H	CAN_H línea del bus dominante (high)
8	-	Reservado (línea de error)
9	CAN_V+	Potencia opcional

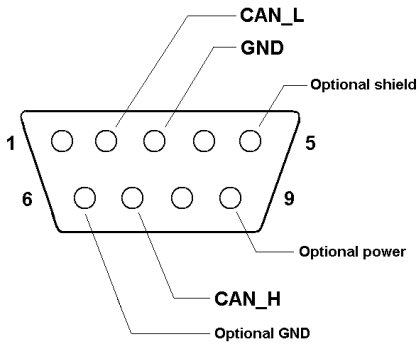


Figura 15. Conector sub D9. Tomado de: https://www.kvaser.com/about-can/the-can-protocol/can-connectors/

7.2.4. Temporización

Cuando se tiene una red de dispositivos CAN se tienen muchos nodos en los cuales no siempre se maneja un mismo valor de reloj por esto es importante sincronizar los nodos ya que el corrimiento de la fase provoca fallos de comunicación.

La sincronización se lleva a cabo cuando un nodo está en la capacidad de prorrogar o cortar el tiempo de duración de un bit para esta operación el protocolo maneja diferentes segmentos de bit divididos en unidades llamadas "time quantas" (TQ), el time quanta es una unidad de tiempo dependiente de la frecuencia de oscilación del reloj; para el cálculo del time quanta se utiliza la siguiente formula y se adjuntan los valores utilizados para este proyecto:

$$t_q = \frac{BRP}{F_{osc}}$$ *Ecuación 1*

$$t_{bit} = \frac{1}{NBR} = 8\,\mu$$ *Ecuación 2*

$$t_q = \frac{2 * (1 + 1)}{4\,MHz} = 1\,\mu s$$

$$NBR = \frac{1}{t_{bit}} = 125\,kb/s$$ *Ecuación 3*

Donde t_q equivale al "time quanta", BRP equivale al valor de la tasa prescalador de baudios que para este proyecto se definió como 4 y y F_{osc} es la frecuencia del oscilador que en este proyecto es de 4 MHz debido a conveniencia puesto que no es necesario que el oscilador funcione a mayor frecuencia ya que la transmisión de datos es de 125 kbits/s dando como resultado un tiempo de "time quanta" de: 1 μs

Para los segmentos del bit se tienen las siguientes secciones:

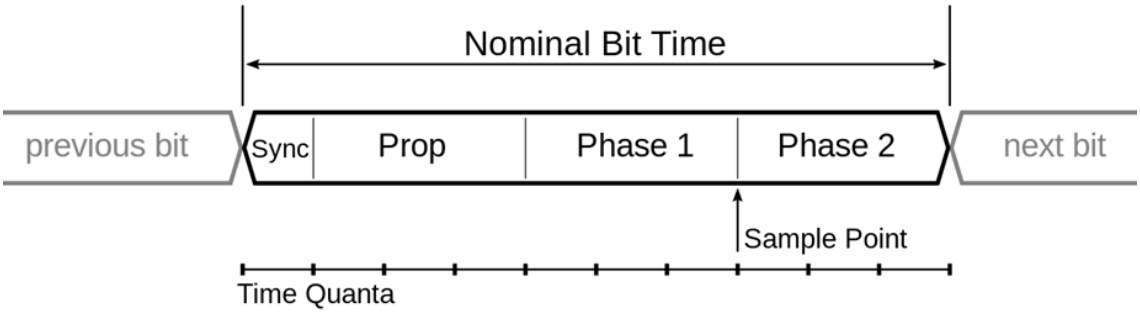


Figura 16. Segmentos de division del bit. Tomado de: https://es.wikipedia.org/wiki/Bus_CAN#/media/File:CAN_Bit_Timing2.svg

De la figura anterior cabe resaltar que el bit se divide en 4 segmentos que son:

Sincronización

Básicamente este segmento realiza una sincronización de los relojes de cada uno de los nodos, generalmente se usa este segmento para realizar una transición entre bits y normalmente se le asigna un curso de 1 "time quanta"

Propagación

Se usa este segmento para proveer de un tiempo para que la información del bus pueda llegar a cada uno de los nodos y se le puede asignar un curso que va desde 1 a 8 "time quanta" y este valor tiene que ser el doble de la adición máxima de la duración de propagación en los nodos más alejados.

Phase 1

Se usa este segmento para prolongar el tiempo del bit y se le puede asignar un curso de 1 hasta 8 "time quanta"

Phase 2

Se usa este segmento para cortar el tiempo del bit y se le puede asignar un curso de 1 hasta 8 "time quanta"

7.2.5. Formas de tramas en CAN

Para la transmisión de datos el protocolo CAN maneja cuatro modelos de tramas en la red que son las siguientes:

Data frame

La data frame (trama de datos) es la trama que se utiliza normalmente para transmitir la información de un nodo a otro, se divide en múltiples campos los cuales son:

- Start of frame

En este segmento reside un bit dominante, el cual se encarga de indicarle a los nodos que se está dando el inicio de la transmisión de información ya que el intercambio de información se puede dar si y solo si el bus está vacío para evitar colisión de datos, básicamente este bit principalmente realiza una sincronización de los nodos dentro de la red.

- Campo de arbitraje

En este segmento reside el identificador de la información transmitida, adicionalmente el identificador realiza una decisión en el momento que colisionen los datos debido a que este identificador contiene la prioridad de cada paquete de información. Este segmento también contiene el bit RTR que es el encargado de informar si la trama contenida es de datos o remota, este bit RTR tiene que ser dominante para que se indique una trama que contiene datos.

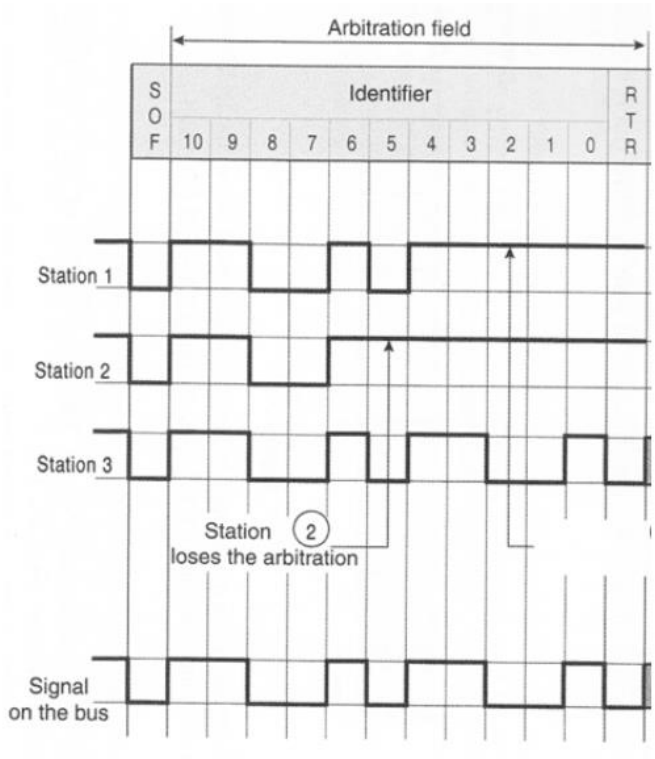


Figura 17. Segmento campo de arbitraje. Tomado de: http://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/pacheco_h_je/capitulo2.pdf

- Campo de datos

En este segmento se ubican los datos propiamente y se compone de una cantidad de bytes que pueden ir de 0 a 8 bytes, el primer bit que se trasmite es el MSB (Most significant bit) como se muestra en la figura:

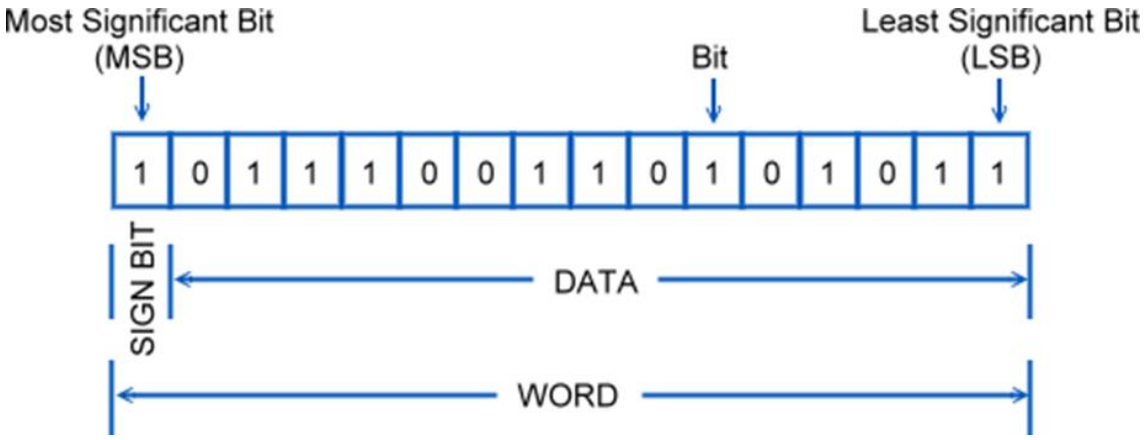


Figura 18. Campo de datos. Tomado de: http://www.plcdev.com/book/export/html/67

- Campo CRC**:**

Este segmento contiene un código cíclico de redundancia (CRC), este código es construido en el emisor y esta adjunta al mensaje para que luego el receptor vuelva a construir este código para compararlo, si se encuentra una discrepancia automáticamente se etiqueta como un mensaje con error. Este segmento también contiene el CRC delimitador que consta de un solo bit de carácter recesivo que posibilita que todos los nodos respondan a los bits previos lo que significa que es en este periodo el nodo valida el CRC.

La función principal del CRC es permitir que la información sea enviada por un canal ruidoso, para esta función el emisor crea un código de verificación el cual es función de la información y, por otro lado, el receptor utiliza esta misma función para volver a determinar el código de verificación y compararlos entre sí para establecer si la información fue correctamente recibida.

- Campo de reconocimiento

Este segmento consta de 2 bits que son los siguientes:

- ✓ ACK slot

  Este bit actúa como verificador de la trama de información recibida, este bit es enviado por el transmisor en forma de bit recesivo y aunque solo un nodo reciba la información sin errores, sobrescribe el bit ACK slot en forma dominante.

- ✓ ACK delimiter

  Este bit siempre es un bit recesivo ya que cuando la información ha sido recibida correctamente por todos los nodos de la red, el bit ACK slots se rodea de dos bits recesivos

- Campo de reconocimiento

Finalmente, la trama de datos finaliza con una flag (bandera) que consta de una serie de siete bits en forma recesiva, y es un campo fijo el cual indica que la trama de datos finaliza.

A continuación, se presenta una imagen en la que se resume la forma de una trama para el protocolo CAN:

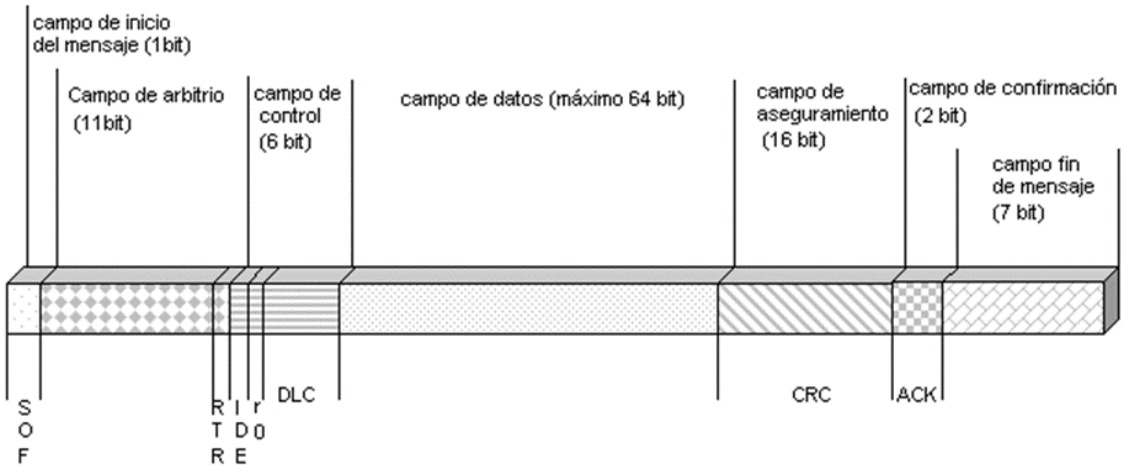


Figura 19. Estructura de la trama. Tomada de: https://www.researchgate.net/figure/Figura-1-Estructura-del-bus-CAN-32-211-Estructuras-de-mensaje-CAN-Los-paquetes-de_fig1_277018667

Remote frame

Los nodos dentro de la red CAN envían información sin advertir si esta ha sido conveniente para alguno de los nodos, por eso en algunos casos se puede presentar que un nodo requiera información de cierto tipo para poder realizar una tarea específica y no cuente con esta información, en estos casos es necesario que el nodo sin la información requerida inicie la solicitud de transmisión de información procedente de otro nodo, dirigiendo una remote frame (trama remota).

La trama remota también se compone de seis campos a diferencia de los siete de la trama de datos, los campos de la remote frame son:

- Start of frame

Este segmento es igual que el segmento de start of frame en la trama de datos

- Campo de arbitraje

En este segmento se realiza el arbitraje al igual que en la trama de datos y contiene dos bits que son el bit identificador y un bit inmediato (RTR).

En cuanto al bit identificador es igual que en la trama de datos, pero el bit RTR para este caso se encuentra en estado recesivo en la trama remota a diferencia de la trama de datos

- Campo de datos

El campo de datos no existe, pero es necesario que se especifiquen la longitud de los datos, en este espacio solo se adjunta la información del tamaño de los datos que serán enviados.

En cuanto al campo CRC, de reconocimiento y el de fin de trama, todos estos segmentos son iguales que en la trama de datos.

En resumen, la trama remota se usa para pedir información de otro nodo, su estructura es casi igual a la trama de datos, pero con la diferencia de que el campo de datos no existe y este espacio se usa para especificar el tamaño de la trama de datos que se está solicitando.

Error frame

La trama de error se utiliza para que cuando la información se recibió de forma errada, esta información se vuelva a enviar. Esta trama se compone de dos campos que son:

- Campo de error flags (Banderas de error)

Este segmento se compone de 6 a 12 bits en los cuales se envían bits dominantes y recesivos en los cuales se ubican las banderas de error.

- Campo delimitador de error

Este compone de 8 bits todos recesivos que permiten aislar el error de la demás información

Overload frame

Esta trama funciona proveyendo un tiempo de desfase que ayuda a que el nodo receptor tenga más tiempo para poder procesar la última información que recibió, este tiempo de desfase es igual a la longitud de la trama de sobrecarga.

Esta trama se compone de seis bits dominantes y se envían en el segmento entre las tramas, los nodos receptores solo pueden pedir máximo dos tramas de sobre carga en forma continua, intermitentemente pueden pedir un número ilimitado.

7.3. Desarrollo dispositivo

En la actualidad existen múltiples dispositivos que se encargan del manejo y control del protocolo CAN, el mayor inconveniente con estos dispositivos es que son muy costosos por lo que la mejor alternativa para la creación del dispositivito descrito en este proyecto es el uso de circuitos que integran un microcontrolador con modulo CAN embebido.

7.3.1. Selección de circuito

Para la selección del circuito se consultaron múltiples proyectos en los que se observó un factor común, siempre los circuitos que integraban un

microcontrolador con modulo CAN se componían de dos elementos, el microcontrolador y el transceptor.

A continuación, se presenta la función que cumple cada uno de los componentes dentro del circuito

Microcontrolador

El microcontrolador como ya se había descrito anteriormente es un circuito integrado en sí, este es el encargado de recibir las señales del protocolo RS-232 por medio del módulo USART y transformarlas a las señales dentro los parámetros regidos por el protocolo CAN.

Transceptor

El transceptor es el encargado de recibir la señal de salida del microcontrolador y llevar esta señal a los niveles de tensión propios del protocolo CAN.

7.3.2. Selección de microcontrolador

Actualmente hay diferentes fabricantes de microcontroladores, los más comunes son de la marca Microchip technology, Atmel, NXP Semiconductors, entre otros, sin embargo, los dispositivos de la marca Microchip son generalmente usados para este tipo de proyectos y aplicaciones, por lo cual, de esta marca es el microcontrolador usado.

Microchip fabrica múltiples tipos de microcontroladores según requerimientos, estos se clasifican por familia, capacidad, tamaño y serie. Los únicos microcontroladores con modulo CAN son los de la familia de gama alta y de la serie 18, 24, 30 y 32 de los cuales se descartan los de la serie 30 y 32 ya que son para el desarrollo de herramientas software de protocolo CAN

A continuación, se presentan los tres candidatos que según especificaciones se acercan a los objetivos de este proyecto, dos son de la serie 18 y uno de la serie 24

- PIC18F2580
- PIC18F2480
- PIC24HJ128

Para la función a cumplir dentro del proyecto se tienen parámetros que se deben ponderar con respecto a las cualidades de cada microcontrolador siendo las más destacables: capacidad de memoria (20%), módulos de comunicación (30%), y costo del componente (50%), siendo este último componente el de mayor ponderación ya que el resto son de similar prestaciones y no se requieren altos valores de memoria y los módulos se hace trascendental que posean módulo USART y módulo ECAN por lo que los tres cumplen esto, sin embargo la

variación de la puntuación depende de otros parámetros como la facilidad de uso de estos módulos y la finalidad de esta arquitectura.

La calificación que se asignara a cada ítem es de una escala de 1 a 5 donde 1 representa la calificación más deficiente y 5 la calificación más excelente

Tabla 3. Matriz de decisión de la selección del microcontrolador

Microcontrolador	Capacidad de Memoria (20%)	Módulos de Comunicación (30%)	Costo (50%)	Resultado
PIC18F2480	3	4	4	3.8
PIC18F2580	4	5	4	4.3
PIC24HJ128	4	5	2	3.3

Según la matriz de decisión previamente realizada, el microcontrolador más apropiado para este proyecto es el Microchip PIC18F2580, ya que posee características que hacen que sus prestaciones sean las más apropiadas para este proyecto con relación al costo.

Figura 20.PIC18F2580. Tomado de: https://www.microchip.com/wwwproducts/en/PIC18F2580

A continuación, se presenta una descripción breve de las características más relevantes del microcontrolador escogido:

- Posee tecnología ECAN
- 10-Bit A/D
- Tecnología NanoWatt

Adicionalmente el PIC18F2580 se destaca por estar dentro de los dispositivos con arquitectura Harvard que le posibilita un acceso a la memoria de datos y a la memoria de instrucciones de manera paralela e individual, también tiene memorias independientes que hacen que cada una tenga capacidad y anchos

diferentes, la memoria de datos de un ancho de un byte y la memoria de instrucciones de 16 bits.

- Frecuencia de operación: DC-40MHz
- Memoria de programa (Bytes): 32768
- Memoria de programa (Instrucciones): 16384
- Memoria de datos (Bytes): 1536
- Memoria EEPROM de datos (Bytes): 256
- Fuentes de interrupciones: 19
- Puentes de entrada y salida: A, B, C, (E)
- Timers: 4
- Módulos PWM: 1
- Módulos ECAN: 1
- Comunicaciones seriales: MSSSP, Enhanced USART
- Set de instrucciones: 75 instrucciones, 87 con el set de instrucciones extendido habilitado

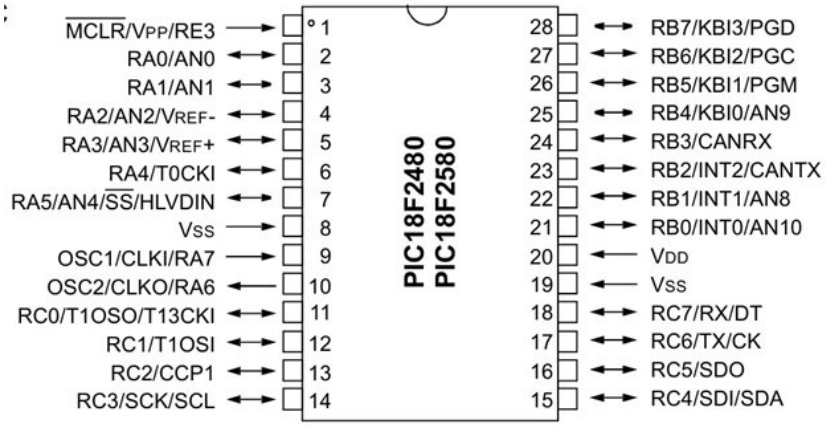


Figura 21. Pines PIC18F2580. Tomado de: http://www.go-gddq.com/html/s225/2012-03/909951.htm

Tabla 4. Descripción de pines

PUERTO	Entrada/Salida	Designación	Descripción
MCLR/Vpp/RE3	I	MCLR	Reinicio, este pin reinicia el microcontrolador
		Vpp	Entrada de voltaje para programación
		RE3	Entrada general
RA0/AN0	I/O	RA0	E/S puerto general

		AN0	A/D Entrada del canal 0
RA1/AN1	I/O	RA1	Entrada general
		AN1	A/D Entrada del canal 1
RA2/AN2/Vref-	I/O	RA2	Entrada general
		AN2	A/D Entrada del canal 2
		Vref-	Entrada de voltaje negativo de referencia
RA3/AN3/Vref+	I/O	RA3	Entrada general
		AN3	A/D Entrada del canal 3
		Vref+	Entrada de voltaje positivo de referencia
RA4/T0CKI	I/O	RA4	Entrada general
		T0CKI	Reloj de temporización para T0
RA5/AN4/SS/HLVDIN	I/O	RA5	Entrada general
		AN4	A/D Entrada del canal 4
		SS	Modulo SPI
		HLVDIN	
VSS	I	VSS	Ground
OSC1/CLKFI/RA7	I	OSC1	Oscilador de cristal
		CLKFI	Reloj externo
		RA7	Entrada general
OSC2/CLKO/RA6	O	OSC2	Oscilador de cristal
		CLKO	Salida Fosc/4
		RA6	Entrada general
RC0/T1OSO/T13CKI	I/O	RC0	Entrada general
		T1OSO	Salida oscilador
		T13CKI	Reloj de temporización para T1
RC1/T1OSI	I/O	RC1	Entrada general
		T1OSI	Entrada oscilador
RC2/CCP1	I/O	RC2	Entrada general

		CCP1	E/S PWM0
RC3/SCK/SCL	I/O	RC3	Entrada general
		SCK	E/S reloj del modulo
		SCL	E/S reloj del modulo
RC4/SDI/SDA	I/O	RC4	Entrada general
		SDI	Entrada de datos
		SDA	E/S de datos
RC5/SDO	I/O	RC5	Entrada general
		SDO	Salida del modulo
RC6/TX/CK	I/O	RC6	Entrada general
		TX	Salida asíncrona USART
		CK	Reloj asíncrono USART
RC7/RX/DT	I/O	RC7	Entrada general
		RX	Entrada asíncrona USART
		DT	Datos USART asíncronos
Vss	I	VSS	Tierra
Vdd	I	VDD	Voltaje positivo
RB0/INT0/AN10	I/O	RB0	Entrada general
		INT0	Interrupción externa
		AN10	A/D Entrada del canal 10
RB1/INT1/AN8	I/O	RB1	Entrada general
		INT1	Interrupción externa
		AN8	A/D Entrada del canal 8
RB2/INT2/CANTX	I/O	RB2	Entrada general
		INT2	Interrupción externa
		CANTX	Salida CAN
RB3/CANRX	I/O	RB3	Entrada general
		CANRX	Entrada CAN
RB4/KBI0/AN9	I/O	RB4	Entrada general
		KBI0	-

		AN9	A/D entrada del canal 9
RB5/KBI1/PGM	I/O	RB5	Entrada general
		KBI1	-
		PGM	Programación del microcontrolador
RB6/KBI2/PGC	I/O	RB6	Entrada general
		KB12	-
		PGC	Programación del microcontrolador
RB7/KBI3/PGD	I/O	RB7	Entrada general
		KBI3	-
		PGD	Programación del microcontrolador

7.3.3. Selección del transceptor

El transceptor CAN es un elemento trascendental dentro del circuito ya que completa el proceso del microcontrolador y lleva la señal de salida a niveles de tensión propios del protocolo CAN. Múltiples fabricantes de microcontroladores también fabrican este tipo de transceptores por lo que en esta selección se presentaron 4 diferentes transceptores para realizar la selección de este y son los siguientes:

- Texas instruments SN65HVD233-HT
- NXP PCA82C250
- Microchip MPC2551
- Onsemi NCV7341

Para la selección del transceptor se tuvo en cuenta la precisión de la transformación de la seña de salida con una ponderación del 30%, compatibilidad en la interfaz con una ponderación del 10% y el costo con una ponderación del 60%.

La calificación que se asignara a cada ítem es de una escala de 1 a 5 donde 1 representa la calificación más deficiente y 5 la calificación más excelente

Tabla 5. Matriz de decisión de la selección del transceptor

Transceptor	Precisión (30%)	Interfaz (10%)	Costo (60%)	Resultados
SN65HVD233-HT	4	3	2	2.7
PCA82C250	5	4	4	4.3
MPC255	5	3	2	3
NCV7341	3	3	5	4.2

Según la matriz de decisión previamente realizada, el transceptor más apropiado para este proyecto es el NXP PCA82C250, ya que posee características que hacen que sus prestaciones sean las más apropiadas para este proyecto con relación al costo.

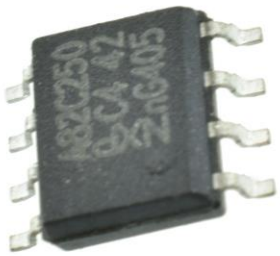


Figura 22.Transceptor NXP PCA82C250. Tomado de: https://www.nxp.com/docs/en/data-sheet/PCA82C250.pdf

A continuación, se presentan las características más relevantes del transceptor escogido

- Totalmente compatible con la norma ISO11989
- Soporta altas velocidades (más de 1 MBd)
- Resistente a los ambientes ruidosos
- Control de pendientes para reducir la interferencia de radio frecuencia (RFI)
- Térmicamente protegido
- Pueden ser conectados hasta 110 nodos

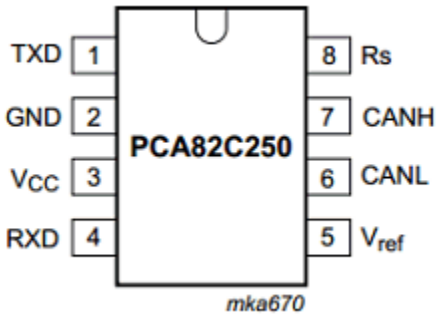


Figura 23. Pines transceptor PCA82C250. Tomado de: http://www.datasheetcafe.com/pca82c250-datasheet-pdf/

Tabla 6. Descripción de pines transceptor

Puerto	Pin	Descripción
TXD	1	Entrada de datos de transmisión
GND	2	Tierra
Vcc	3	Entrada de voltaje
RXD	4	Salida de datos de recepción
Rs	5	Salida de voltaje de referencia
CANH	6	Low level CAN voltaje entrada/salida
CANL	7	High level CAN voltaje entrada/salida
Vref	8	Entrada de resistor

7.3.4. Programación del microcontrolador

Para la programación del microcontrolador, se tuvieron en cuenta 2 programaciones distintas, si bien este proyecto sólo se centra en el desarrollo del dispositivo de enlace entre los equipos del banco de aviónica de la universidad y el bus CAN (de RS-232 a CAN), para la prueba de comunicación se usaron 2 computadores transmitiendo por protocolo RS-232 emulando los dispositivos del banco de aviónica ya que no fue posible tener acceso a los equipos, así que es necesario tener una programación también para enlazar nuevamente a un equipo que verifique la presencia de la señal en el bus que en este caso será otro computador (de CAN a RS-232)

Para llevar a cabo la programación se necesitaron múltiples herramientas que son:

- MPLAB

Es un editor especial para los productos de Microchip basado en un sistema de ventanas, el subíndice IDE se refiere al Integrated Development Enviroment por las siglas en inglés, esto significa que su funcionamiento es bajo un solo ambiente que integra todas las funciones para poder desarrollar el código de funcionamiento del microcontrolador, además permite acceder a las configuraciones de cada una de las librerías previamente programadas para los microcontroladores de esta marca y permite grabar el microcontrolador con el código desarrollado por el programador en el módulo.

- Proteus

Es un software de simulación de circuitos para validar de forma virtual su funcionamiento adicionalmente esta herramienta puede ser enlazado con MPLAB para verificar el funcionamiento del código de programación del microcontrolador.

- PICKIT

El PICKIT es un dispositivo especializado en la depuración y programación de microcontroladores de la marca Microchip PIC, que tiene que ser usado en conjunto con el editor MPLAB

Figura 24. PICKIT 4. Tomado de: https://www.robotshop.com/us/es/depurador-en-circuito-mplab-pickit-4.html

- Compilador XC8

El compilador XC8, cuenta con múltiples registros pregrabados que conceden acceso a los múltiples módulos físicos y de software del microcontrolador como los osciladores internos, PWM, timers, etc. Esto presenta una mayor facilidad a la hora de realizar la programación ya que el programador puede ahorrar tiempo porque estos accesos están programados en el lenguaje c y se tiene mayores ventajas con respecto al lenguaje ensamblador permitiéndole mayor enfoque el desarrollo del proyecto al programador.

7.3.4.1. Simulación del microcontrolador en Proteus

En el software de Proteus se llevó a cabo sólo la simulación del funcionamiento del circuito y la sección de código de programación de la comunicación serial ya que este software no soporta elementos de verificación del protocolo CAN. Este software se utilizó con el objetivo de validar el código sin tener que llevar a cabo una prueba real debido a que la memoria de programación del microcontrolador tiene un número limitado de ciclos de reescritura.

7.3.4.2. Bits de configuración del microcontrolador

Los modos de configuración son unas preconfiguraciones que permiten activar o desactivar diferentes modos de funcionamiento del microcontrolador según los requerimientos que se tengan, esto permite administrar los recursos del microcontrolador.

Los modos de configuración se describirán a continuación tanto para el microcontrolador del dispositivo maestro (el cual cumple la función de recibir la información en protocolo RS-232 y enviarla por protocolo CAN) como para el microcontrolador del dispositivo esclavo (el cual cumple la función de recibir la información en protocolo CAN y enviarla por protocolo RS-232) así en esta tesis sólo se hubiera planteado necesidad del primero ya que para la prueba de comunicación son necesarios los dos dispositivos.

Se presentará el código que genera MPLAB describiendo los modos de configuración, este código se genera automáticamente cuando se administran los recursos del microcontrolador, sólo es necesario especificar cómo se quiere que se usen estos recursos

7.3.4.2.1. Modo de configuración dispositivo maestro

Estos son las configuraciones del microcontrolador del dispositivo maestro

#pragma config OSC = IRCIO67 Se le indica que se quiere que funcione con el oscilador interno del microcontrolador, si se quiere que funcione con un oscilador externo es necesario especificarlo.

#pragma config FCMEN = OFF Se le indica que el bit de habilitación para el Fail-Safe clock monitor este desactivado

#pragma config IESO = OFF Se le indica que el bit para el Internal/External Oscillator Switchover este desactivado

#pragma config PWRT = OFF Se le indica que el bit de habilitación para el Power-up Timer este desactivado

#pragma config BOREN = BOHW Se le indica que el bit de habilitación para el Brown-out Reset este habilitado

#pragma config BORV = 3 Se le indica que el bit para el Brown-out Reset Voltage este sea 3V

#pragma config WDT = OFF Se le indica que el bit de habilitación para el Watchdog Timer este desactivado

#pragma config WDTPS = 32768 Se le indica que los bits de habilitación para el Watchdog Timer Postscale sea 1:32768

#pragma config PBADEN = ON Se le indica que los bits de habilitación para el PORTB A/D para entradas analógicas en el reseteo

#pragma config LPT1OSC = OFF Se le indica que los bits de habilitación para el Low-Power Timer 1 Oscillator este desactivado

#pragma config MCLRE = OFF Se le indica que los bits de habilitación para el MCLR Pin este desactivado, el pin RE3 está habilitado

*#pragma config STVREN = ON*Se le indica que los bits de habilitación para el Stack Full/Underflow Reset para entradas analógicas en el reseteo

#pragma config LVP = OFF Se le indica que los bits de habilitación para el Single-Supply ICSP este desactivado

#pragma config BBSIZ = 1024 Se le indica que los bits para el Boot Block Size Select sea 1k

#pragma config XINST = OFF Se le indica que los bits de habilitación para el Extended Instruction Set este desactivado

#pragma config CP0 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CP1 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CP2 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CP3 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CPB = OFF Se le indica que los bits de habilitación para el Boot Block Code Protection este desactivado

#pragma config CPD = OFF Se le indica que los bits de habilitación para el Data EEPROM Code Protection este desactivado

#pragma config WRT0 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRT1 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRT2 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRT3 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRTC = OFF Se le indica que los bits para el Configuration Register Write Protection este desactivado

#pragma config WRTB = OFF Se le indica que los bits para el Boot Block Write Protection este desactivado

#pragma config EBTR0 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTR1 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTR2 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTR3 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTRB = OFF Se le indica que los bits para el Boot Block Table Read Protection este desactivado

7.3.4.2.2. Modo de configuración dispositivo esclavo

Estos son las configuraciones del microcontrolador del dispositivo esclavo

#pragma config OSC = IRCIO67 Se le indica que se quiere que funcione con el oscilador interno del microcontrolador, si se quiere que funcione con un oscilador externo es necesario especificarlo.

#pragma config FCMEN = OFF Se le indica que el bit de habilitación para el Fail-Safe clock monitor este desactivado

#pragma config IESO = OFF Se le indica que el bit para el Internal/External Oscillator Switchover este desactivado

#pragma config PWRT = OFF Se le indica que el bit de habilitación para el Power-up Timer este desactivado

#pragma config BOREN = BOHW Se le indica que el bit de habilitación para el Brown-out Reset este habilitado

#pragma config BORV = 3 Se le indica que el bit para el Brown-out Reset Voltage este sea 3V

#pragma config WDT = OFF Se le indica que el bit de habilitación para el Watchdog Timer este desactivado

#pragma config WDTPS = 32768 Se le indica que los bit de habilitación para el Watchdog Timer Postscale sea 1:32768

#pragma config PBADEN = ON Se le indica que los bit de habilitación para el PORTB A/D para entradas analógicas en el reseteo

#pragma config LPT1OSC = OFF Se le indica que los bits de habilitación para el Low-Power Timer 1 Oscillator este desactivado

#pragma config MCLRE = OFF Se le indica que los bits de habilitación para el MCLR Pin este desactivado, el pin RE3 está habilitado

#pragma config STVREN = ON Se le indica que los bits de habilitación para el Stack Full/Underflow Reset para entradas analógicas en el reseteo

#pragma config LVP = OFF Se le indica que los bits de habilitación para el Single-Supply ICSP este desactivado

#pragma config BBSIZ = 1024 Se le indica que los bits para el Boot Block Size Select sea 1k

#pragma config XINST = OFF Se le indica que los bits de habilitación para el Extended Instruction Set este desactivado

#pragma config CP0 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CP1 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CP2 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CP3 = OFF Se le indica que los bits de habilitación para el Code Protection este desactivado

#pragma config CPB = OFF Se le indica que los bits de habilitación para el Boot Block Code Protection este desactivado

#pragma config CPD = OFF Se le indica que los bits de habilitación para el Data EEPROM Code Protection este desactivado

#pragma config WRT0 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRT1 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRT2 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRT3 = OFF Se le indica que los bits para el Write Protection Code Protection este desactivado

#pragma config WRTC = OFF Se le indica que los bits para el Configuration Register Write Protection este desactivado

#pragma config WRTB = OFF Se le indica que los bits para el Boot Block Write Protection este desactivado

#pragma config EBTR0 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTR1 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTR2 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTR3 = OFF Se le indica que los bits para el Table Read Protection este desactivado

#pragma config EBTRB = OFF Se le indica que los bits para el Boot Block Table Read Protection este desactivado

7.3.4.3. Módulos usados del microcontrolador maestro

7.3.4.3.1. Iniciación Modulo USART

A continuación, se presentan los fragmentos de código comentado donde se presentan instrucciones relacionadas con el módulo USART.

```
TRISCbits.TRISC7=1;//pin RX Como una entrada digital

TRISCbits.TRISC6=0;//pin TX Como una salida digital

TXSTA=0b00100110;//8bits, transmisión, asíncrono, alta velocidad

RCSTA=0b10010000;// USART PIC, recepción 8 bits, asíncrono

SPBRG=25;//para 9600baudios con un oscilador de 4Mhz
```

7.3.4.3.2. Iniciación del módulo CAN

A continuación, se presentan los fragmentos de código comentado donde se presentan instrucciones relacionadas con el módulo ECAN

```
TRISBbits.RB2=1; //puertos modulo CAN

TRISBbits.RB3=1;
```

```
CANCON=0b10000000;//inicializa Configuration Mode
while(!(CANSTAT & 0b10000000));// espera
ECANCON=0b01000000;// inicia modo 0
BRGCON1=0b00000000;// Baud rate
BRGCON2=0b10001010;
BRGCON3=0b01000001;
CANCON=0;//Inicializa Normal mode
```

7.3.4.4. Módulos usados del microcontrolador esclavo

7.3.4.4.1. Iniciación Modulo USART

```
TRISCbits.TRISC7=1;//pin RX como una entrada digital
TRISCbits.TRISC6=0;//pin TX como una salida digital
TXSTA=0b00100110;//8bits, transmisión, asíncrono, alta velocidad
RCSTA=0b10010000;// USART PIC, recepción 8 bits, asíncrono
SPBRG=25;//para 9600baudios con un oscilador de 4Mhz
```

7.3.4.4.1.1. Iniciación del módulo CAN

```
TRISBbits.RB2=1;//puertos modulo CAN
TRISBbits.RB3=1;
CANCON=0b10000000;//inicializa Configuration Mode
while(!(CANSTAT & 0b10000000));// espera
ECANCON=0b01010010;// inicia modo 0 y define bufer 0 de recepción
BRGCON1=0b00000000;// Baud rate
BRGCON2=0b10001010;
BRGCON3=0b01000001;
RXF0SIDH=0B11111111;//filtros buffer 0
RXF0SIDL=0B11100000;//filtro estándar
RXM0SIDH=0B11111111;//máscara buffer 0
RXM0SIDL=0B11100000;//máscara estándar
RXFCON0bits.RXF0EN=1;//activa filtros
```

RXFBCON0=0b00100010;//asocia filtro 0 con buffer 0

CANCON=0;//Inicializa Normal mode

while (CANSTAT & 0xE0); // espera

7.3.4.5. Interrupciones

Una interrupción es la capacidad que tiene un microcontrolador de ejecutar un programa como respuesta ante un acontecimiento externo, cuando una interrupción ocurre el microcontrolador pausa el programa que este ejecutando, ejecutar el programa de la interrupción y luego retorna nuevamente a lo que se encontraba haciendo. Para este proyecto fue fundamental usar interrupciones en el dispositivo maestro (que es el centro del proyecto) ya que sin ellas no es posible pasar la información de un módulo a otro.

7.3.4.5.1. Interrupciones de alta prioridad

void __interrupt (high_priority) interrupcion2 (void){//rutina de interrupción por transmisión CAN

if(PIR3bits.TXB0IF==1){//si la bandera se pone a 1

if(data.d6==1){//Si el bit 6 del char se pone en 1

TXB0SIDL=0B11100000;//Identificador estándar

TXB0SIDH=0B11111111;//Identificador estándar

TXB0D0=data.dato;//se da valor al bufer de transmisión 0

TXB0DLC=0B00000001;//Se transmite 1 byte

TXB0CONbits.TXREQ=1;//Se activa la transmisión

__delay_ms(10);//pausa de 10 ms

PIE3bits.TXB0IE=0; //Desactiva la interrupción de transmisión CAN

}

else

if(data.d6==0){ //Si el bit 6 del char se pone en 0

TXB1SIDL=0B11100000;//Identificador estándar

TXB1SIDH=0b11100100;//Identificador estándar

TXB1D0=data.dato;//se da valor al bufer de transmisión 1

TXB1DLC=0B00000001;//Se transmite 1 byteh

TXB1CONbits.TXREQ=1;//Se activa la transmisión

```
    __delay_ms(10);//pausa de 10 ms
    PIE3bits.TXB0IE=0; //Desactiva la interrupción de transmisión CAN
    }
}
```

7.3.4.5.2. Interrupciones de baja prioridad

```
void __interrupt (low_priority) interrupcion (void){//rutina de interrupción por recepción USART
    if(PIR1bits.RCIF==1){//si la bandera se pone a 1
    data.dato=RCREG;//Lee registro de recepción USART
    PIE3bits.TXB0IE=1; //Activa interrupción y bandera transmisión CAN
    TXBIEbits.TXB0IE=1;
    BIE0bits.B0IE=1;
    PIR3bits.TXB0IF=1;
    }
    }
```

7.3.4.6. Código comentado completo dispositivo maestro

```
#define _XTAL_FREQ 4000000 // define la frecuencia del oscilador a 4MHz
#include <xc.h>
#include "uart.h" // llamar header usart
union{
struct{
   unsigned d0:1; //nombre bit:tamaño
   unsigned d1:1;
   unsigned d2:1;
   unsigned d3:1;
   unsigned d4:1;
   unsigned d5:1;
   unsigned d6:1;
   unsigned d7:1;
```

```
};
unsigned char dato; //nombre y tipo dato
}data;//nombre de la unión

//data.dato para char completo
//data.dx para bit

void main(void) {

   OSCCONbits.IRCF2=1;
   OSCCONbits.IRCF1=1;
   OSCCONbits.IRCF0=0;// define la frecuencia del oscilador a 4MHz

    __delay_ms(5);

   RCONbits.IPEN=1;//activa interrupciones por periféricos

   INTCON=0b11000000;//habilita el uso de interrupciones

   IPR3bits.TXB0IP=1;//alta prioridad transmisión
   IPR1bits.RCIP=0;//baja prioridad recepcion

   iniciar_usart();//inicia el módulo USART

   TRISBbits.RB2=1; //puertos modulo CAN
   TRISBbits.RB3=1;
   CANCON=0b10000000;//inicializa Configuration Mode
   while(!(CANSTAT & 0b10000000));// espera
```

```
   ECANCON=0b01000000;// inicia modo 0
   BRGCON1=0b00000000;// Baud rate
   BRGCON2=0b10001010;
   BRGCON3=0b01000001;
   CANCON=0;//Inicializa Normal mode
   PIE1bits.RCIE=1;//Activa interrupción de recepción USART
   return;
}

void __interrupt (low_priority) interrupcion (void){//rutina de interrupción por recepción USART

    if(PIR1bits.RCIF==1){//si la bandera se pone a 1

    data.dato=RCREG;//Lee registro de recepción USART

    PIE3bits.TXB0IE=1; //Activa interrupción y bandera de transmisión CAN
    TXBIEbits.TXB0IE=1;
    BIE0bits.B0IE=1;
    PIR3bits.TXB0IF=1;

    }
   }

 void __interrupt (high_priority) interrupcion2 (void){//rutina de interrupción por transmisión CAN

if(PIR3bits.TXB0IF==1){//si la bandera se pone a 1
   if(data.d6==1){//Si el bit 6 del char se pone en 1
```

```
   TXB0SIDL=0B11100000;//Identificador estándar
   TXB0SIDH=0B11111111;//Identificador estándar
   TXB0D0=data.dato;//se da valor al bufer de transmisión 0
   TXB0DLC=0B00000001;//Se transmite 1 byte
   TXB0CONbits.TXREQ=1;//Se activa la transmisión
   __delay_ms(10);//pausa de 10 ms

   PIE3bits.TXB0IE=0; //Desactiva la interrupción de transmisión CAN
   }
   else
   if(data.d6==0){ //Si el bit 6 del char se pone en 0
   TXB1SIDL=0B11100000;//Identificador estándar
   TXB1SIDH=0b11100100;//Identificador estándar
   TXB1D0=data.dato;//se da valor al bufer de transmisión 1
   TXB1DLC=0B00000001;//Se transmite 1 byteh
   TXB1CONbits.TXREQ=1;//Se activa la transmisión
   __delay_ms(10);//pausa de 10 ms

   PIE3bits.TXB0IE=0;//Desactiva la interrupción de transmisión CAN
   }
}
}
```

7.3.4.7. Código comentado completo dispositivo esclavo

```
#include <xc.h>
#include <stdio.h>
#include "uart.h"
#define _XTAL_FREQ 4000000 // define la frecuencia del oscilador a 4MHz
```

```
unsigned char dato;

void main(void)
{
   OSCCONbits.IRCF2=1;
   OSCCONbits.IRCF1=1;
   OSCCONbits.IRCF0=0;// define la frecuencia del oscilador a 4MHz

   TRISBbits.RB2=1;//puertos modulo CAN
   TRISBbits.RB3=1;
   CANCON=0b10000000;//inicializa Configuration Mode
   while(!(CANSTAT & 0b10000000));// espera
   ECANCON=0b01010010;// inicia modo 0 y define bufer 0 de recepción
   BRGCON1=0b00000000;// Baud rate
   BRGCON2=0b10001010;
   BRGCON3=0b01000001;
   RXF0SIDH=0B11111111;//filtros buffer 0
   RXF0SIDL=0B11100000;//filtro estandar
   RXM0SIDH=0B11111111;//máscara buffer 0
   RXM0SIDL=0B11100000;//máscara estándar
   RXFCON0bits.RXF0EN=1;//activa filtros
   RXFBCON0=0b00100010;//asocia filtro 0 con buffer 0
   CANCON=0;//Inicializa Normal mode
   while (CANSTAT & 0xE0); // espera
   iniciar_usart();//inicia modulo USART

while(1){
```

```
BSEL0bits.B0TXEN=0;//recepción automática
B0SIDL=0B11100000;//buffer estándar
B0SIDH=0B11111111;
dato=B0D0;//lectura del buffer de recepción
B0DLC=0B00000001;//recibe 1 byte

if(B0CONbits.RXFUL==1){//si la recepción es exitosa

TXREG=dato;//transmite el byte de información por el módulo USART
B0CONbits.RXFUL=0;//se admite un nuevo mensaje

}
}
}
```

7.3.5. Prueba de comunicación

Uno de los pilares del proyecto es la verificación y validación del dispositivo, ya que permite saber si este cumple con los objetivos del proyecto y puede ser implementado, esta validación y verificación fue llevada a cabo a través de una prueba de comunicación que permitió conocer el comportamiento del dispositivo.

Esta prueba de comunicación se llevó a cabo con tres computadoras que debían ser interconectadas mediante el bus CAN, en medio del bus y la computadora se ubicaba el dispositivo ya que así se emulaba su funcionamiento a la salida de los equipos del banco de aviónica de la universidad, uno de los dispositivos funcionaba como dispositivo maestro y dos como dispositivos esclavos verificando que la información fuera discriminada según la pertinencia y la programación de los identificadores, a continuación se realiza la explicación de cada una de las condiciones:

- Dispositivo maestro:

Emisor de los caracteres generador de un identificador que sirve para que la información sea aceptada o rechazada por el dispositivo receptor

- Dispositivo esclavo

Receptor de los caracteres, es el encargado de discriminar el identificador que genera el dispositivo maestro.

Es importante definir algunos términos que se presentaran a continuación para que sea más clara la forma en que funcionan los identificadores:

Carácter: Es una unidad de información que corresponde a letra, números y signos de puntuación.

Bit: Es un digito del sistema de numeración binario.

Para la asignación del identificador y los filtros se tuvo en cuenta el sexto bit del carácter, en la siguiente disposición:

- Dispositivo maestro:

Si el sexto bit es un 1, el identificador estándar del mensaje es 11111111111 y si el sexto bit es un 0, el identificador estándar del mensaje es 11100100111

- Dispositivo esclavo:

El primer esclavo estaba configurado para recibir el mensaje solo si el identificador era 11111111111, de lo contrario lo rechazaba.

El segundo esclavo recibía el mensaje si el identificador era 11100100111, de lo contrario lo rechazaba

Figura 25. Configuración de la prueba de comunicación

Se enviaría un mensaje compuesto por números y letras a través de la computadora con dispositivo maestro, una de las computadoras esclavo discriminaría los datos con números y admitiría los datos de letras mientras que la otra computadora con dispositivo esclavo discriminaría los datos con letras y admitiría los datos con números ya que el sexto bit de las letras es 1 y de los números es 0 según el código ASCII.

Carácter	ASCII	Carácter	ASCII
A	0100 0001	W	0101 0111
B	0100 0010	X	0101 1000
C	0100 0011	Y	0101 1001
D	0100 0100	Z	0101 1010
E	0100 0101	0	0011 0000
F	0100 0110	1	0011 0001
G	0100 0111	2	0011 0010
H	0100 1000	3	0011 0011
I	0100 1001	4	0011 0100
J	0100 1010	5	0011 0101
K	0100 1011	6	0011 0110
L	0100 1100	7	0011 0111
M	0100 1101	8	0011 1000
N	0100 1110	9	0011 1001
O	0100 1111	+	0010 1011
P	0101 0000	-	0010 1101
Q	0101 0001	*	0010 1010
R	0101 0010	:	0011 1010
S	0101 0011	=	0011 1101
T	0101 0100	<	0011 1100
U	0101 0101	;	0011 1011
V	0101 0110		

Tabla 1.2. Código ASCII

Figura 26. Codigo ASCII

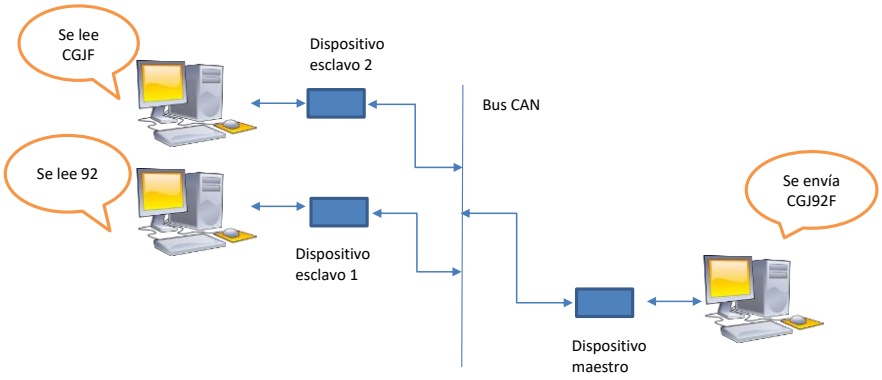


Figura 27. Esquema de prueba de comunicación

Así se probaría que el dispositivo se encarga de la transformación del protocolo y que es capaz de discriminar la información que le es pertinente a cada nodo.

Se descartó la idea de llevar a cabo la prueba con una interfaz de controlador CAN ya que su costo era demasiado elevado y pude ascender alrededor de 7 millones de pesos.

7.3.5.1. Selección de terminales emuladoras

Los equipos del banco de aviónica no estaban disponibles, por lo que fue necesario emular estos equipos utilizando computadoras y un puerto conversor de USB a RS-232.

7.3.5.2. Cable de USB a RS-232

Se utilizó un cable comercial para transformar la comunicación USB a protocolo RS-232 ya que las computadoras no poseen puertos para comunicación serial, adicionalmente para poder escribir sobre el terminal USB se utilizó un software de código abierto y gratuito llamado PUTTY.

Figura 28. Logo de PUTTY. Tomado de: https://www.redbubble.com/es/people/zurgr/works/25938205-putty-logo?p=art-print

7.3.5.3. Evaluación de éxito de la prueba

Para cuantificar el éxito de la prueba de comunicación se plantearon estos ítems de evaluación con una respectiva calificación de 1 a 5 donde 1 representa la calificación más deficiente y 5 la calificación más excelente

Los ítems de evaluación son:

- Conexión de los nodos
- Envió y recepción de información
- Discriminación de la información
- Aceptación de la información
- Fidelidad de la información

La conexión de los nodos será evaluada teniendo en cuenta si hay comunicación a través de todos los componentes, si los dispositivos tienen una conexión satisfactoria se catalogará como éxito este ítem

El envío y recepción de la información será evaluado teniendo en cuenta si el dispositivo logra recibir la información y enviarla por los módulos correctos, si es así, se catalogará como éxito este ítem

La discriminación de la información será evaluada teniendo en cuenta si el dispositivo rechaza toda información que no le sea pertinente, si es así este ítem será catalogado como exitoso.

La aceptación de la información será evaluada teniendo en cuenta si el dispositivo acepta toda información que le sea pertinente, si es así este ítem será catalogado como exitoso.

La fidelidad de la información será evaluada teniendo en cuenta si la información enviada por el maestro es recibida eficazmente, si la información es recibida completa por el esclavo de manera correcta y ordenada se catalogará este ítem como exitoso

7.3.6. Desarrollo carcasa

El dispositivo tiene como objetivo ser implementado en el banco de aviónica por lo que es necesario que tenga un tamaño compacto y que el circuito este ensamblado de manera tal que no sea susceptible a modificaciones o alteraciones involuntarias así que se realizó el montaje del circuito sobre una PCB (Printed Circuit Board) adicionalmente por su objetivo dentro del banco es necesario integrar un elemento de protección al circuito y sus componentes, creando así un dispositivo integral que cumpla todos los requerimientos.

7.3.6.1. PCB

Como se había mencionado anteriormente la PCB es una tarjeta de circuito impreso que simplifica las conexiones de un circuito y optimiza el espacio generando caminos de comunicación entre los componentes de un circuito.

Para el desarrollo de esta tarjeta se hicieron múltiples intentos por generar el diseño en diferentes softwares especializados en este tipo de labores, pero para la posterior fabricación eran incompatibles y los softwares de uso comercial para estos diseños no son de libre licencia así que se optó contratar a un tercero para el diseño y fabricación de la tarjeta.

Se realizó la PCB tiene las siguientes características:

- Espacio entre pistas, Vias y Pad: 0.2 mm
- Perforación: 0.4 mm
- Drill Annulus: 0.25 mm
- Ancho de pista: 0.2 mm
- Separación de la dimensión al Cobre: 0.2 mm
- Material: Fibra de vidrio

- Color de la máscara de los componentes: Blanco
- Color del antisolder: Verde

NOTA: En los anexos se adjuntará los archivos Gerber con el diseño de la PCB

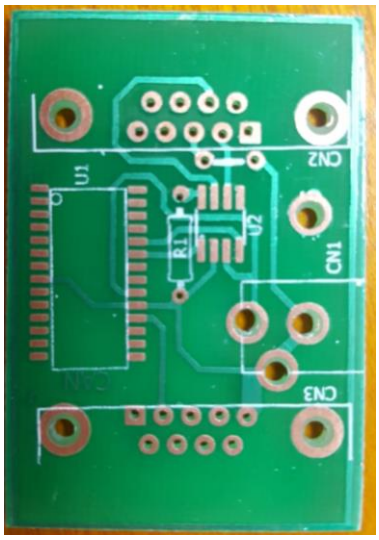


Figura 29. PCB fabricada

7.3.6.2. Carcasa fisca

Para la carcasa se realizó un diseño de protección al circuito y sus componentes teniendo en cuenta la arquitectura de la PCB y sus componentes y que el usuario tuviera fácil acceso a los conectores sin que se viera afectado la protección del circuito, este diseño se realizó en CATIA para posteriormente ser fabricado a través de impresión 3D.

Esta carcasa de protección se diseñó para que se ensamble a través de dos piezas y 4 pernos de sujeción y permitir así un fácil acceso al circuito en caso de requerirlo. Así mismo tiene 3 orificios para el acceso a los conectores, estos orificios se hicieron con un ajuste de aproximadamente 1mm en cada espacio ya que el método de impresión 3D no garantiza las tolerancias impuestas en el diseño.

Durante la fabricación de la carcasa se presento el inconveniente de elegir el material de fabricación ya que hay varios materiales en los cuales se realiza manufactura por impresión 3D pero los mas destacables son el PLA (Ácido Poliláctico) y el ABS (Acrilonitrilo Butadieno Estireno) de los cuales se realiza un cuadro comparativo con las ventajas y desventajas de cada uno

Tabla 7. Ventajas y desventajas de los materiales para la carcasa

PLA	ABS
Ventajas	
Es fabricado a base de maíz	Es reciclable
No expele gases tóxicos al ser fundido	Precio más bajo en comparación con el PLA
Posee una gran variedad de colores	Posee mejor resistencia, flexibilidad y durabilidad
Desventajas	
No es reciclable	Cuando se funde produce gases tóxicos
Es más costoso en comparación con el ABS	Derivado del petróleo

Luego de haber consultado las ventajas y desventajas de los materiales se opto por el ABS debido a que su costo es menor, por ser un elemento de fabricación única al ser prototipo no se tiene una afectación en cuanto a materia de salud y medio ambiente además de tener las características de resistencia, flexibilidad y durabilidad requeridas para el proyecto.

A continuación, se presentan algunas características importantes de la carcasa:

- Largo: 61 mm
- Ancho: 60 mm
- Altura: 31 mm
- Material: ABS

NOTA: En los anexos se adjuntan los archivos CAD con las piezas que componen la carcasa, además se adjuntan los planos.

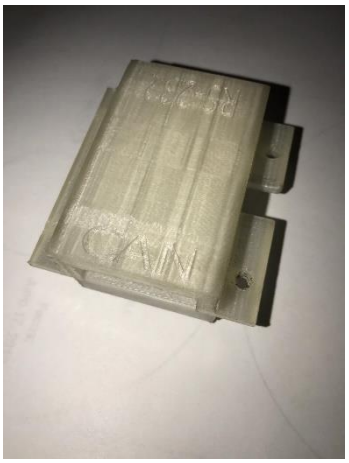


Figura 30. Diseño de carcasa

8. RESULTADOS

8.1. Circuito

La simulación en PROTEUS de la comunicación por el módulo USART permitió observar que esta comunicación a través del microcontrolador era exitosa, se enviaban los caracteres de un terminal a otro, observando como llegaba la información completa y sin errores

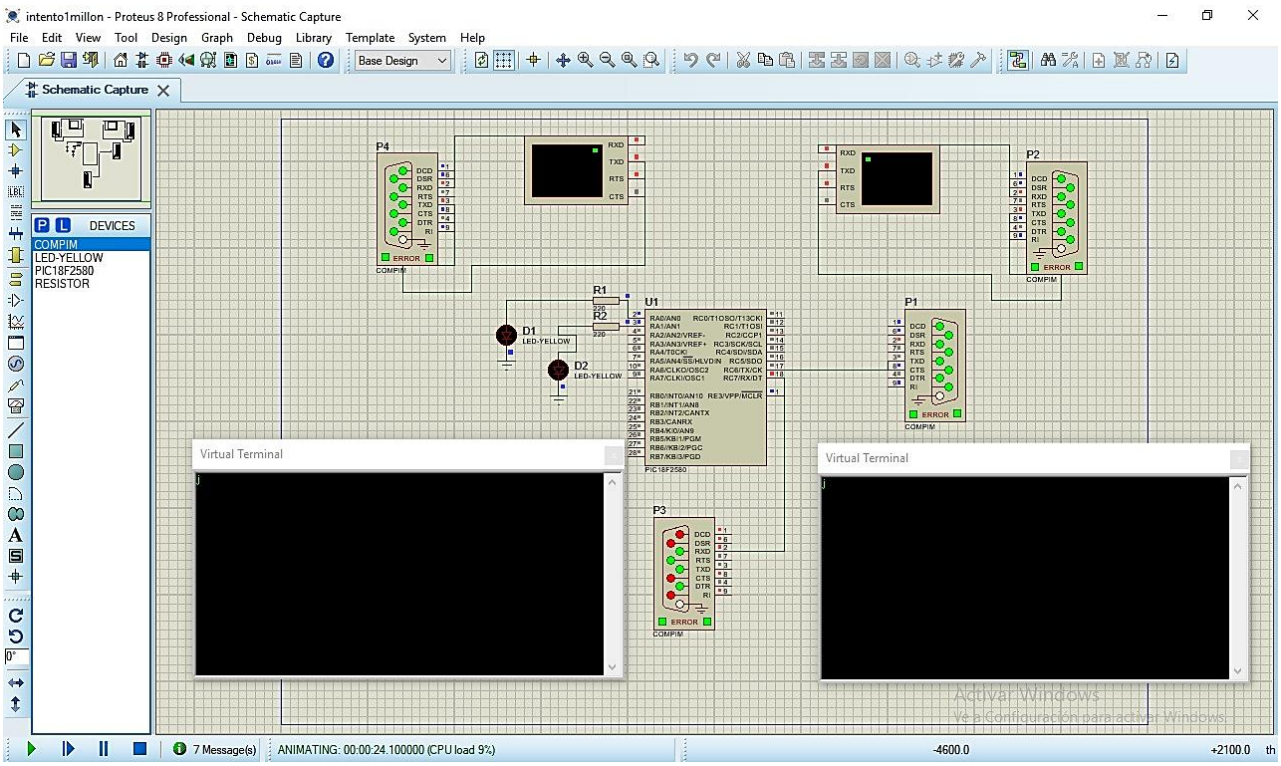


Figura 31. Simulación en PROTEUS del circuito

El circuito final no requirió optoacoplador conteniendo los siguientes componentes:

- Microcontrolador PIC18F2580
- Transceptor PCA82C250
- Puertos DB9 90 grados hembra
- Power Jack 5v 1.5 mm
- Resistencia de 4.7k Ohm

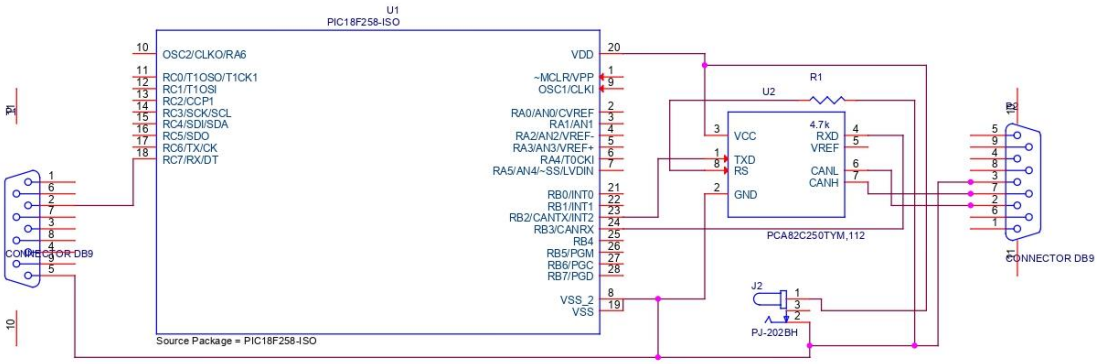


Figura 32. Circuito final

8.2. Dispositivo final (todo ensamblado)

A continuación, se presenta una secuencia de ensamble con el proceso de ensamblado final del dispositivo de enlace, cabe resaltar que las piezas fueron diseñadas para ser acopladas de una forma sencilla y en la cual el usuario puede identificar fácilmente la correcta posición de las piezas sin cometer errores.

1. Disposición de la carcasa

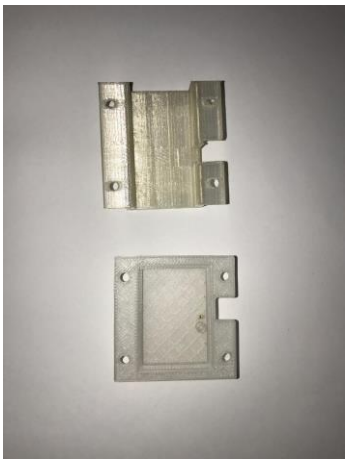


Figura 33. Disposición de la carcasa

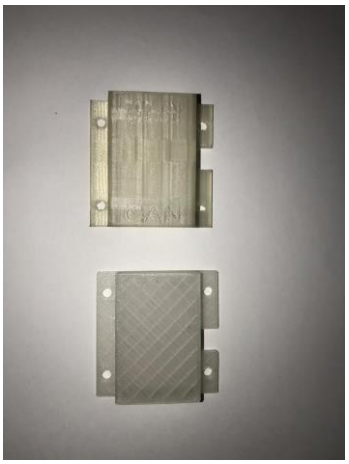


Figura 34. Disposición de la carcasa

2. Unión PCB con pieza carcasa inferior

Figura 35. Unión de PCB con carcasa inferior

3. Unión de las dos piezas carcasa

Figura 36. Unión completa de la carcasa

Figura 37. Unión de la carcasa

Figura 38. Unión de la carcasa

4. Sujeción de las dos piezas carcasa por medio de tornillos

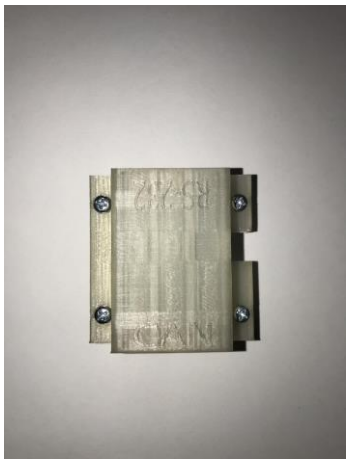


Figura 39. Sujeción de la carcasa

Figura 40. Dispositivo conectado

8.3. Resultados de prueba

La prueba de comunicación se realizó con una estructura de 3 computadoras, una actuando como dispositivo maestro y las otras dos como dispositivos esclavos como se había mencionado anteriormente.

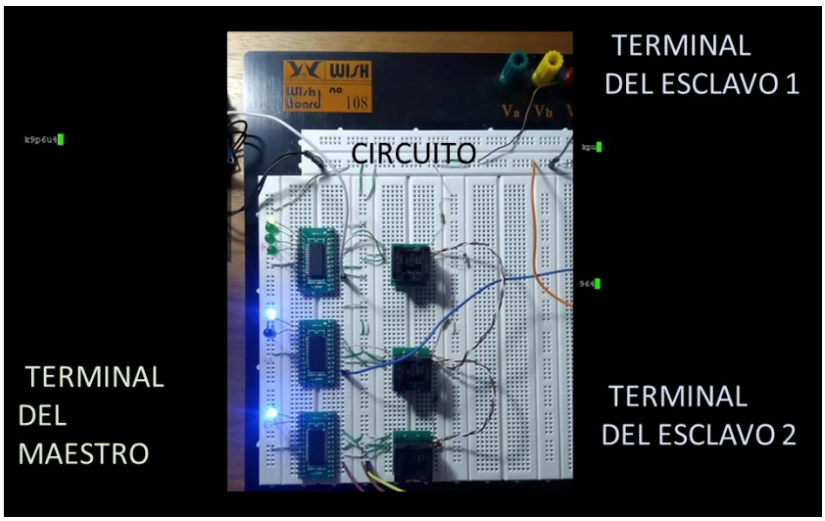


Figura 41. Configuración prueba de comunicación

Como se observa en la figura anterior, cada uno de los dispositivos tenían una serie de LEDs los cuales funcionaban como testigos de las funciones, módulos e interrupciones del microcontrolador que se querían monitorear además ver qué pasaba con la información y el comportamiento del microcontrolador ante ella.

A continuación, se describe la función de monitoreo de cada uno de los LEDs en el dispositivo maestro (LEDs verdes)

- LED 1: Inicio

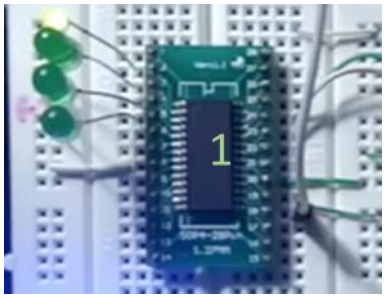


Figura 42. LED inicio

- LED 2: Indica el salto a la interrupción de recepción del módulo USART

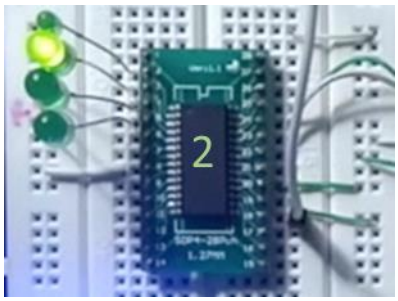


Figura 43. LED interrupción de recepción USART

- LED 3: Indica el salto a la interrupción de transmisión del módulo CAN

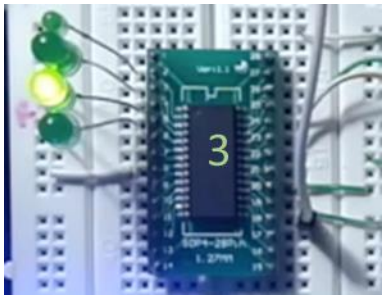


Figura 44. LED interrupción de recepción CAN

- LED 4: Indica transmisión de CAN exitosa

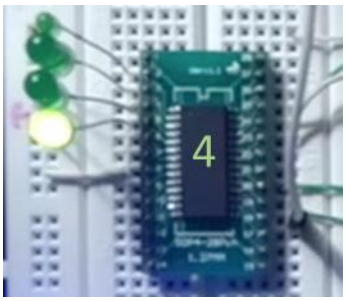


Figura 45. LED Transmisión CAN exitosa

A continuación, se describe la función de monitoreo de cada uno de los LEDs en el dispositivo esclavo (LEDs azul)

- LED1: Inicio

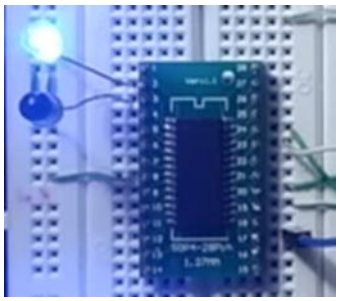


Figura 46. LED inicio dispositivo esclavo

- LED2: Indica recepción CAN exitosa

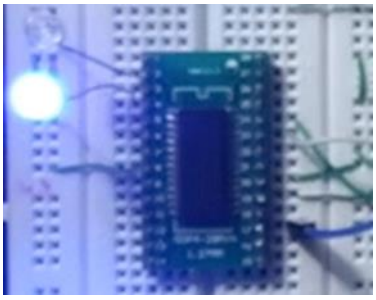


Figura 47. LED recepción CAN exitosa

Para evaluar si la prueba fue exitosa o fallida se realizó una evaluación con los ítems previamente estipulados, con lo cual se le asigno una calificación dada por el tutor del proyecto.

Tabla 8. Evaluación de la prueba de comunicación

Item	Descripción	Puntaje	Aprobado por el tutor
Éxito de comunicación	Los dispositivos se comunican entre si	**5**	**SI**
Envió y recepción de información	Los dispositivos envían información a través de los módulos	**5**	**SI**
Discriminación de la información	Los dispositivos rechazan la información que no necesitan	**5**	**SI**
Aceptación de la información	Los dispositivos solo aceptan la información que necesitan	**5**	**SI**
Fidelidad de la información	La información es correcta	**5**	**SI**

8.4. Recomendaciones

Debido a que no se tuvo acceso a los equipos del banco de aviónica se hacen las siguientes recomendaciones para ser tenidas en cuenta a la hora de realizar el acople del dispositivo tanto a los equipos que se encuentran actualmente como los que se añadirán posteriormente

Para Maestro con salida en rs-232

- Se recomienda que los datos sean enviados en paquetes de 8 bits (1 byte) en los cuales los primeros tres o cuatro bits puedan funcionar como identificadores, esto depende directamente del valor requerido a enviar, puesto que este caso permite máximo el envío del valor 31 en decimal, en el caso de que el valor decimal requerido sea mayor a 31 se debe hacer uso de 2 bytes, utilizando un byte como identificador y el otro para el almacenamiento del dato a enviar; para realizar esta acción se debe configurar el módulo USART del microcontrolador usado en el conversor para que reciba cadenas de caracteres usando arreglos para el almacenamiento de la cadena, el módulo CAN no necesita alteraciones a menos de que se requiera el envío de más de un byte de información.

- Se recomienda que el puerto de salida del dispositivo sea DB9 hembra para que se pueda hacer uso de un cable DB9 macho-macho para realizar las conexiones entre el pin TX del dispositivo y el RX del conversor.

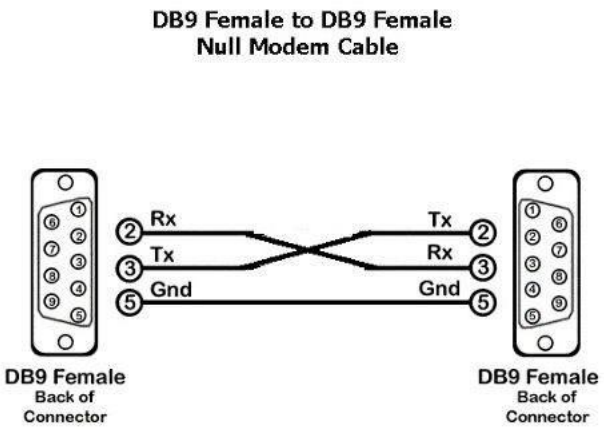


Figura 48. Conexión esperada entre dispositivo y conversor

- Se recomienda verificar la velocidad de envío de datos del dispositivo, en este caso el conversor está configurado para una recepción de datos USART de 9600 baudios, pero esto está sujeto a cambios según el diseño del dispositivo maestro que envía datos en serial RS-232

Para el BUS CAN

- Se recomienda que se utilicen un par de cables trenzados donde se conecten los niveles de CAN HIGH y CAN LOW junto con otro cable que conecte la tierra de todos los nodos puesto que esta debe ser común entre los nodos CAN para que la comunicación funcione de forma adecuada

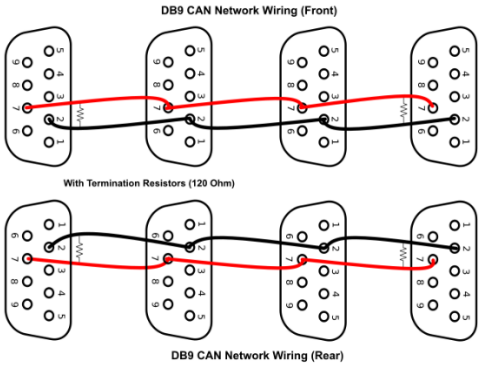
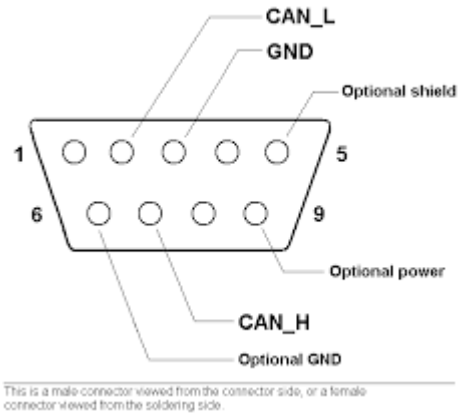


Figura 49. Bus de comunicacion CAN y conector DB) para protocolo CAN



Para el dispositivo conversor

- Se podría llegar a realizar una modificación en el dispositivo conversor en la cual la alimentación se realice del nodo maestro que envía datos en serial RS-232; como no se tiene claro si el nodo maestro puede realizar esta operación se prefirió alimentar de forma independiente el conversor.

9. CRONOGRAMA DE ACTIVIDADES

ID	TAREA	DURACION	INICIO	FIN	OCTUBRE	NOVIEMBRE	DICIEMBRE	ENERO	FEBRERO	MARZO	ABRIL	MAYO
1	Objetivo 1	28 días	24/10/2018	30/11/2018								
2	Tarea 1A	28 días	24/10/2018	30/11/2018								
3	Tarea 1B	28 días	24/10/2018	30/11/2018								
4	Tarea 1C	28 días	24/10/2018	30/11/2018								
5	Objetivo 2	117 días	1/11/2018	12/04/2019								
6	Tarea 2A	23 días	1/11/2019	3/12/2018								
7	Tarea 2B	21 días	1/02/2019	1/03/2019								
8	Tarea 2C	40 días	18/02/2019	12/04/2019								
9	Objetivo 3	40 días	18/03/2018	10/05/2019								
10	Tarea 3A	40 días	18/03/2018	10/05/2019								
11	Objetivo 4	12 días	15/04/2019	30/04/2019								
12	Tarea 4A	5 días	15/04/2019	19/04/2019								
13	Tarea 4B	3 días	22/04/2019	24/04/2019								
14	Tarea 4C	4 días	25/04/2019	30/04/2019								

10. PRESUPUESTO

Dispositivo	Cantidad	Valor
TAR MAX232 DB9	3	$36 495.60
6N137	6	$46 548.52
Shipping	1	$19 395.22
Cargo mínimo pedido	1	$11 637.13
PIC18F2580	7	$117 740.36
PCA82C250	7	$36 736.82
PICKit 4	1	$295 000
Adaptador 300	2	$20 000
Adaptador Sep 28	2	$7 000
Flete	1	$61 023
Cable conversor USB-RS232	3	$29 705
Conector Jack hembra	1	$274
Conector DB9 Hembra	2	$2 185
Diseño y construcción de PCB	1	$87 000
Impresión 3D carcasa	1	$57 000
	Total:	$827 740.65

11. CONCLUSIONES

Es necesario realizar reparaciones y actualizaciones a los equipos del banco de aviónica puesto que la correcta operación del dispositivo se basa en la forma del funcionamiento de estos equipos.

Los equipos que se añadirán posteriormente al banco de aviónica de la universidad deben contar con los requerimientos indicados en las recomendaciones debido que así se garantiza una integración exitosa en materia de comunicación con los demás equipos del banco de aviónica.

Para el desarrollo de este tipo de dispositivos a nivel académico que manejan protocolo CAN, es altamente recomendable usar microcontroladores Microchip PIC, en especial de la familia 18 ya que su efectividad es bastante alta con relación al precio, mientras que para usos industriales se recomienda el uso de controladores CAN prefabricados y especializados ya que son especialmente diseñados para administrar el protocolo, aunque su precio es elevado.

En el circuito generalmente se utiliza un optoacoplador el cual termina de refinar la señal del protocolo CAN, este no es necesario para el circuito y se comprobó que se puede obtener un resultado satisfactorio sin la implementación de este ya que los sistemas optoacoplados generalmente se utilizan para ambientes con mucha interferencia a causa del ruido.

Para la programación del dispositivo se comprobó que son indispensables las interrupciones ya que permiten pasar de un módulo a otro dentro microcontrolador, las máscaras y filtros igualmente permiten que la información sea discriminada y sin una correcta configuración de estos ítems, el microcontrolador no podrá operar bajo los parámetros requeridos.

La PCB (Printed Circuit Board) es un elemento que permite reducir el tamaño, y cableado dentro del circuito, sin embargo si no se pretende realizar una fabricación en masa, el costo es muy elevado, se recomienda que para dispositivos experimentales los montajes sean realizados en protoboard o en baquelita.

12. BIBLIOGRAFIA

Almeida, L., Pedreiras, P., & Fonseca, J. A. G. (2002). The FTT-CAN protocol: Why and how. *IEEE Transactions on Industrial Electronics*, *49*(6), 1189–1201. https://doi.org/10.1109/TIE.2002.804967

Cely, Allain; Ojeda, Martiza; Jiménez, J. (2008). *DISEÑO DE UN LABORATORIO DIDACTICO DE AVIONICA PARA LA UNIVERSIDAD SAN BUENAVENTURA*.

Encinas, D., Frati, F. E., & Naiouf, M. (2011). Caracterización del comportamiento de una red de sensores basado en COTS bajo condiciones de temperatura hostil para sistemas aeroespaciales Caracterización del comportamiento de una red de sensores basado en COTS bajo condiciones de temperatura hostil para sistemas aeroespaciales, (October).

Encinas, D., & Meilan, P. (2009). Protocolo de comunicaciones CAN aplicado a sistemas satelitales y vehículos lanzadores. *XV Congreso Argentino …*, (May 2015). Retrieved from http://sedici.unlp.edu.ar/handle/10915/21218

Fernando, O., & Ortiz, P. (2016). Modelamiento y simulación de una subred ATN tierra-tierra, para aplicaciones AIDC (Aeronautical Interfacility Data Communications) y AMHS (Aeronautical Message Handling System) aplicando la normatividad y recomendaciones de OACI, para el soporte de servic.

Historia del transistor. (2007). *Tecnología Hecha Palabra*. Retrieved from http://www.tecnologiahechapalabra.com/tecnologia/genesis/articulo.asp?i=1793

Rao, T. . (2009). CAN BUS IN AVIATION. Retrieved from https://www.aviationtoday.com/2009/05/01/can-bus-in-aviation/

Microchip Technology Inc. (2009). *microchip*. Obtenido de microchip: http://ww1.microchip.com/downloads/en/devicedoc/39637d.pdf

Guerrero, C. (2019). DISEÑO Y CONSTRUCCIÓN DE UN ALTÍMETRO CON DISPLAY DIGITAL. En L. F. Guerrero, *DISEÑO Y CONSTRUCCIÓN DE UN ALTÍMETRO CON DISPLAY DIGITAL* . Bogotá: Universidad de San Buenaventura, Bogotá.

Guevara, O. (2012). *avecomputointe*. Obtenido de avecomputointe: http://avecomputointe.blogspot.com/2012/05/interrupciones-en-microcontroladores.html

Hernández, P. (2011). Monitoreo de hábitos de manejo por medio de una red CAN automotriz. En J. E. Hernández, *Monitoreo de hábitos de manejo por medio de una red CAN automotriz.* Cholula, Puebla, México: Universidad de las americas, Puebla.

Ibrahim, D. (2008). *Advanced PIC Microcontroller Projects in C, from USB to RTOS with the PIC18F Series.* Elsevier.

Semiconductors, N. (2011). *PCA82C250*. Obtenido de PCA82C250: https://www.nxp.com/docs/en/data-sheet/PCA82C250.pdf

13. ANEXOS

- Archivos Gerbers de la PCB (CD)
- Archivos CATPART de CATIA de la carcasa de protección (CD)
- Video de prueba de comunicación (CD)
- Archivos de programación en MPLAB (CD)
- Planos de la carcasa de protección