SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

INSTITUTO TECNOLÓGICO DE MÉRIDA

TESIS

"ANALISIS DE LA RED FULL CAN EN EL DISEÑO DE SISTEMAS DE CONTROL A PARTIR DE DATOS EXPERIMENTALES"

PARA OPTAR AL GRADO DE:

MAESTRO EN INGENIERIA

PRESENTA:

ING. JUAN ALBERTO OJEDA ARANA

ASESOR:

DR. CARLOS LUJAN RAMIREZ

MÉRIDA, YUCATÁN, MÉXICO.

02 DE DICIEMBRE DE 2016

SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de Mérida

DEPENDENCIA: DIV. DE EST. DE POSG. E INV.
No. DE OFICIO: X-562/2016

ASUNTO: AUTORIZACIÓN DE IMPRESIÓN

Merida, Yucatán A 16 de noviembre de 2016

C. JUAN ALBERTO OJEDA ARANA
PASANTE DE MAESTRIA EN INGENIERÍA

De acuerdo al fallo emitido por su asesor el **Dr. Carlos Alberto Luján Ramírez**, co-dirigido por el **Dr. Jesús Sandoval Gio** y la comisión revisora integrada por el Dr. Agustín Alfonso Flores Novelo, y el Dr. José Ramón Atoche Enseñat, considerando que cubre los requisitos establecidos en el Reglamento de Titulación de los Institutos Tecnológicos le autorizamos la impresión de su trabajo profesional con la TESIS:

"ANÁLISIS DE LA RED FULL CAN EN EL DISEÑO DE SISTEMAS DE CONTROL A PARTIR DE DATOS EXPERIMENTALES"

ATENTAMENTE
IN HOC SIGNO VINCES

M.C. MIRIAM M. SÁNCHEZ MONROY
JEFA DE LA DIVISIÓN DE ESTUDIOS
DE POSGRADO E INVESTIGACIÓN

S. E. P.
INSTITUTO TECNOLOGICO
DE MERIDA
DIVISION DE ESTUDIOS DE
POSGRADO E INVESTIGACION

C.p. Archivo
MHSM/fjaa.

AGRADECIMIENTOS.

Agradezco a Dios por todas las bendiciones,

A mi familia por su apoyo durante el trayecto de mi carrera,

A los maestros de la Maestría en ingeniería por el tiempo y apoyo.

A todas aquellas personas que hicieron posible este trabajo.

El principio de la sabiduría es el temor de Jehová;
los insensatos desprecian la sabiduría y la enseñanza.
Proverbios 1:7

Contenido

INDICE DE TABLAS.

ÍNDICE DE FIGURAS.

Capítulo 1 Presentación.

Introducción.

Desde la aparición del protocolo de comunicación CAN a finales de los años 80s, ha ido evolucionando gracias al avance tecnológico, específicamente en el área de los circuitos integrados. Esto permitió la inclusión de este bus de comunicación a varios ámbitos laborales, tales como: la industrial, la domótica, la biomédica, la petrolera, entre otros.

El bus de comunicaciones CAN ha tomado mucha fuerza en sector industrial, debido a sus reducidos costos de implementación y su fiabilidad en ambientes de mucha interferencia, comunicando diferentes dispositivos para el control de los procesos industriales; por lo tanto, es importante que el bus de comunicaciones siempre esté disponible para que los dispositivos conectados en la red funcionen de forma adecuada.

.

Este trabajo contiene el análisis y diseño de sistemas control en una red con implementación Full CAN de varios nodos.

Planteamiento del problema.

EL bus de comunicaciones CAN (Control Area Network) es un bus de campo que puede ser clasificado de acuerdo a ciertas características, una de ellas es por medio de la configuración del Mailbox. Dicha configuración puede tener las implementaciones: Basic CAN, Full CAN, FIFO, Enhaced Full CAN [1]. La implementación más típica en el bus, en especial en los SCD (Sistemas de Control Distribuido), es el Basic CAN [2], esto es debido a que la mayoría de los módulos CAN, sobre todo, los denominados Stand-Alone tienen embebido esa implementación. Con el paso del tiempo han salido al mercado, nuevos Chips que integran diferentes implementaciones CAN, entre ellos, el Full CAN.

Dicho lo anterior se desea observar y analizar la disponibilidad del bus implementando una red Full CAN en los SCD.

Para realizar lo anterior, se desea caracterizar varios sistemas de control, y con ello, diseñar el control apropiado para cada sistema, mediante la identificación de sistemas con datos experimentales e implementar las técnicas de control disparadas por eventos expuestas en [3].

Objetivo general.

Se diseñarán y analizarán 2 sistemas de control en una red con implementación Full CAN de alta velocidad de transferencia, basados en microcontroladores PSoC5LP, en la cual, un nodo se encargará del procesamientos de los algoritmos de control. Dicho análisis tendrá como parámetros la cantidad de datos transmitidos y el comportamiento de los sistemas de control.

Objetivos específicos.

- Diseñar una PCB para el funcionamiento del transceptor CAN que utilizará el PSOC5LP.
- Definir la red CAN y la tarea de cada nodo.
- Definir los sistemas a controlar.
- Crear una interfaz USB-HID para monitorear la disponibilidad de los nodos en la red CAN y graficar los sistemas de control.
- Desarrollar hardware necesario para el perfecto funcionamiento de los sistemas.
- Obtener la función de transferencia de cada sistema paramétricamente en la red CAN.
- Diseñar e implementar el control para cada sistema planteado.
- Comprobar el estado del bus con el monitor y el impacto en los sistemas.

Justificación.

Desde el año 2002 se ha estado escribiendo artículos sobre los sistemas embebidos críticos en el bus CAN para sistemas automotrices, donde mencionan la importancia de tener un bus siempre disponible para el correcto funcionamiento de los actuadores o módulos del automóvil [4]. Al paso del tiempo el bus de comunicaciones CAN ha ido introduciéndose en diferentes campos, como la domótica. En el 2006 se implementó un bus que interactúa con los dispositivos domésticos básicos [5] pero aún no se había introducido el concepto de los sistemas de control distribuido utilizando el bus CAN.

Con el desarrollo de la tecnología los sistemas embebidos se han introducido diferentes tipos de implementación CAN, como lo es el Basic CAN o el Full CAN [6]. De las cuales existen muchas aplicaciones, en donde predomina la implementación Basic CAN debido a que la mayoría de los integrados solamente tenían incorporada esa implementación. En los SCD, se han aplicado las redes Basic CAN con éxito [2] [7], en cambio, existe muy poca información de las implementaciones Full CAN en sistemas de control. Por lo tanto se desea saber el comportamiento de la misma en un sistema de control.

Por último, existen estudios sobre técnicas de control disparadas por eventos simuladas en el bus CAN (Matlab) [3], dicho estudio, resalta la importancia de aplicar alguno de los algoritmos en una red real.

Estado del arte.

Las comunicaciones digitales han tomado mucha importancia en las últimas décadas, debido al gran crecimiento de la electrónica digital. Es evidente que las comunicaciones digitales han estado sustituyendo a las analógicas debido al menor costo, facilidad de generar y tratar las señales, a esto también se añade el descubrimiento de nuevas técnicas de elaboración de circuitos integrados que permiten que las comunicaciones digitales sean más fáciles.

En el desarrollo de comunicaciones electrónicas encontramos las comunicaciones para uso en telecomunicaciones como el FSK u otro tipo de modulación digital. Para la comunicación entre tarjetas de circuito impreso encontramos las comunicaciones i2c, SPI entre otros. También existen las comunicaciones para un bus de campo de las cuales la más sobresaliente es el bus CAN.

EL bus CAN ha ido evolucionado al paso del tiempo desde su creación a finales de los 80s por R. Bosch. Esta evolución ha permitido al bus CAN su inclusión en diferentes sectores como son la domótica, robótica, industria automotriz, entre otros [1].

Existe mucha información acerca de aplicaciones donde se ha utilizado el bus CAN, implementándolo como Basic CAN (CAN 2.0A) o CAN extendido (CAN 2.0B) de las cuales encontramos lo desarrollado en el 2010 los alumnos de la universidad de Beijín, China [9], donde diseñaron un driver CAN basado en sistemas embebidos, donde utilizaron el modulo CAN de la empresa MICROCHIP MCP2510 con una tarjeta s3c2440, donde el objetivo era lograr una comunicación CAN con módulos independientes. Posteriormente, en ese mismo año la universidad de Tecnología y electrónica de Guilin, China. Implementaron una comunicación basado en tecnologías Sistem On Chip (SOC) cuyo objetivo era realizar una comunicación CAN con un ALTERA cyclone III (FPGA) y realizar un test con un simulador del bus CAN [8]. En el año siguiente, 2011, universitarios de Harbin, Heilongjiang Provincia de China, implementaron una red Basic CAN con el microcontrolador STC89C52 cuyo objetivo es comunicar la red con una Computadora utilizando un adaptador CAN232 [9]. En ese mismo año, alumnos de la universidad politécnica de China

implementaron el bus CAN para un sistema de control distribuido a un motor de un avión utilizando una tarjeta de desarrollo ICTECK-F2812 [2]. Así mismo, alumnos de la universidad de Chongqing, China Desarrollaron un monitor CAN en tiempo real con el objetivo de ver los datos transmitidos en la red, esto es utilizando una tarjeta NI USB-8473 [10].

En el año 2012 en el Instituto Politécnico Nacional se desarrolló una red para instrumentación electrónica basa en el protocolo CAN utilizando microcontroladores dsPIC33FJ128MC804 donde logró realizar una comunicación CAN de 666.66Kbs en 100 metros de distancia [11]. En el año 2013 alumnos de la universidad de Shangai, China un método de simulación y testeo de un bus de comunicaciones CAN utilizando CANspider y MATLAB [12]. En ese mismo año el Dr. Francesco Ortix, de Milan, Italia, escribe como se utilizó el bus de comunicaciones CAN en un cohete espacial utilizando el Ipodout (interfaz), dando importancia en el uso del control distribuido utilizando el bus CAN [13]. En el año 2014, en India se desarrolló un sistema de adquisición de datos basado en el bus CAN y un ARM [14]. En ese mismo año se desarrolló un sistema de tiempo real para operar un avión a escala utilizando el bus CAN en la cual le agregaron una capa de aplicación más (TTCAN), donde resaltaron la importancia de los sistemas distribuidos en el bus [7].En ese mismo año alumnos de Hang Zhou,China aplicaron el bus CAN para el monitoreo y control de perforaciones en aguas profundas utilizando STM32F103VCT6 [15]. En ese mismo año se implementó una comunicación Basic CAN en un dsPIC30F, con los transceptores MCP2550. Dicha red manipulaba motores de DC mediante PWM [16]. En el año 2015 se implementó una red Basic CAN que mantenía comunicación con la computadora vía USB-HID, cuyo objetivo era monitorizar la red Basic CAN [17]. Por ultimo alumnos del CINEDET estudian las técnicas de control simulados en el bus de comunicación CAN para los sistemas de control basado por eventos, con el fin de hacer más eficiente el control utilizando el bus [3].

Capítulo 2 MARCO TEÓRICO.

2.1 El bus CAN (Control Area Network).

El protocolo CAN fue desarrollado a finales de los 80s por el alemán R. Bosch Gnbh para solucionar problemas de interconexión y comunicación de dispositivo en los automóviles. El bus CAN es una comunicación serial que fue diseñado para proporcionar una transmisión de datos en ambientes de mucha interferencia como lo es la interferencia electromagnética o las descargas electroestáticas [18] y sobre todo, para poder implementar una comunicación a bajo costo.

2.1.1 Funcionamiento del bus CAN.

Las principales características, elementos, funcionamiento y trama lo menciona [19], cuyas principales características son:

- La información que circula entre las unidades de mando a través de los dos cables (bus) son paquetes de 0 y 1 (bit) con una longitud limitada y con una estructura definida de campos que conforman el mensaje.
- Uno de esos campos actúa de identificador del tipo de dato que se transporta, de la unidad de mando que lo trasmite y de la prioridad para trasmitirlo respecto a otros. El mensaje no va direccionado a ninguna unidad de mando en concreto, cada una de ellas reconocerá mediante este identificador si el mensaje le interesa o no.
- Todas las unidades de mando pueden ser trasmisoras y receptoras, y la cantidad de las mismas abonadas al sistema puede ser variable (dentro de unos límites).
- Si la situación lo exige, una unidad de mando puede solicitar a otra una determinada información mediante uno de los campos del mensaje (trama remota).
- Cualquier unidad de mando introduce un mensaje en el bus con la condición de que esté libre; si otra lo intenta al mismo tiempo, el conflicto se resuelve por la prioridad del mensaje indicado por el identificador del mismo.
- El sistema está dotado de una serie de mecanismos que aseguran que el mensaje es trasmitido y recibido correctamente. Cuando un mensaje presenta un

error, es anulado y vuelto a trasmitir de forma correcta, de la misma forma una unidad de mando con problemas avisa a las demás mediante el propio mensaje, si la situación es irreversible, dicha unidad de mando queda fuera de servicio pero el sistema sigue funcionando.

2.1.2 Modelo OSI del bus CAN.

Se utiliza el modelo OSI (Open System Interconnection) de 7 capas para definir la red CAN, el modelo fue definido para establecer un estándar en la red para que diferentes componentes de diferentes fabricantes puedan comunicarse. La especificación CAN define el bit encoding, timing y la sincronización de la señal transmitida. Las características eléctricas de los transceptores son dadas por la ISO (International Estándar Organization) o la SAE (Society of Automotive Engineers). La capa física es descrita en la Tabla 1. Las capas superiores del modelo OSI no son especificadas para el bus CAN, por lo tanto, los usuarios pueden crear interfaces únicas que cumplen sus requisitos específicos. Por ejemplo, tenemos el Rockwell (Allen−Bradley) DeviceNet, el Honeywell Smart Distributed System] (SDS), el Kvaser CAN Kingdom, el Time Triggered CAN (TTCAN) y el SAE J1939 [21]. En la Figura 2.1 se presenta la clasificación del modelo OSI en la implementación CAN.

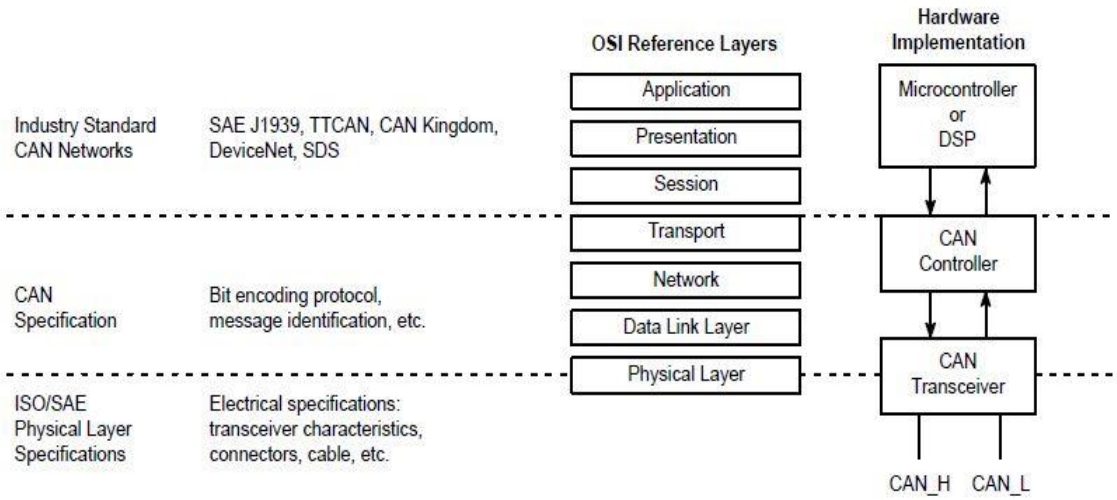

Figura 2.1 Modelo OSI en la implementación CAN.

Tabla 1 Características del bus.

Parameter	High−Speed CAN	Fault Tolerant CAN	Single Wire CAN
Physical Layer Specification	ISO 11898−2	ISO 11519−2, ISO 11898−3, ISO 11992	SAE J2411
Features	High−speed, differential bus, good noise immunity	Ability to detect a wiring error and switch to a single wire mode	Low cost
Popular Applications	Automotive and industrial controls	Large trucks and trailers	Automotive, GM−LAN network
Transmission Speed	1.0 Mbits/s @ 40 meters 125 kbits/s @ 500 meters	125 kbits/s	33.3 kbits/s (normal mode) 83.3 kbits/s (diagnostic mode)
Cable	Twisted or parallel pair wires, shielded or unshielded cable	Twisted or parallel pair wires, shielded or unshielded cable	Single unshielded wire
Termination Resistance	120 Ω resistors located at each end of the bus	Separate CAN_H and CAN_L termination resistors located at each node, resistance determined by number of CAN nodes	Termination resistor located at each node, resistance determined by number of CAN nodes
Min/Max Bus Voltage	12 V System: −3.0/16 V 24 V System: −3.0/32 V	12 V System: −3.0/16 V 24 V System: −3.0/32 V	12 V System: −3.0/16 V
Min/Max Common Mode Bus Voltage	CAN_L: −2.0 (min)/2.5 V (nom) CAN_H: 2.5 (nom)/7.0 V (max)	CAN_L: −2.0 (min)/2.5 V (nom) CAN_H: 2.5 (nom)/7.0 V (max)	CAN_Bus offset voltage = 1.0 V (max)
Transceiver Schematic and Waveform	Figure 7	Figure 8	Figure 9

2.1.3 Elementos que componen el sistema CAN-Bus

Cables

La información circula por dos cables trenzados que unen todas las unidades de control que forman el sistema. Esta información se trasmite por diferencia de tensión entre los dos cables, de forma que un valor alto de tensión representa un 1 y un valor bajo de tensión representa un 0. La combinación adecuada de unos y ceros conforman el mensaje a trasmitir.

En un cable, los valores de tensión oscilan entre 0V y 2.25V, por lo que se denomina cable L (Low); y en el otro, el cable H (High) lo hacen entre 2.75V. y 5V. En caso de que se interrumpa la línea H o que se derive a masa, el sistema trabajará con la señal de Low con respecto a masa; en el caso de que se interrumpa la línea L, ocurrirá lo contrario. Esta situación permite que el sistema siga trabajando con uno de los cables cortados o comunicados a masa, quedando fuera de servicio solamente cuando ambos cables se cortan. Dicho lo anterior, solo es aplicable cuando se implementa la ISO 11519−2, ISO 11898−3, ISO 11992. Por otro lado, cuando se implementa la ISO11898-2, si uno de las líneas del nodo es interrumpido, la comunicación se suspende.

Es importante tener en cuenta que el trenzado entre ambas líneas sirve para anular los campos magnéticos, por lo que no se debe modificar en ningún caso ni el paso ni la longitud de dichos cables. En la **¡Error! No se encuentra el origen de la referencia.**, se representa el cableado y los voltajes de la línea H y L.

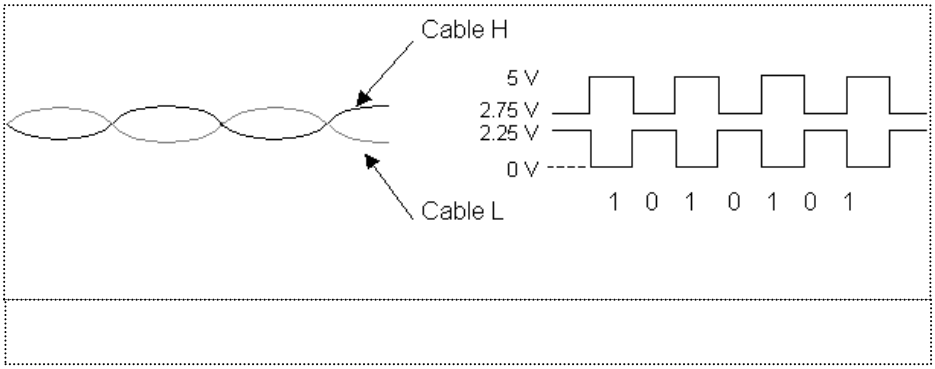

Figura 2.2 Par trenzado CAN y sus valores en voltaje en sus dos estados.

Elemento de cierre o terminador

Son resistencias conectadas a los extremos de los cables H y L. dependen de la velocidad de transmisión de los nodos y permiten adecuar el funcionamiento del sistema a diferentes longitudes de cables y número de unidades de control abonadas, ya que impiden fenómenos de reflexión que pueden perturbar el mensaje. Este elemento es representado en la Figura 2.3:

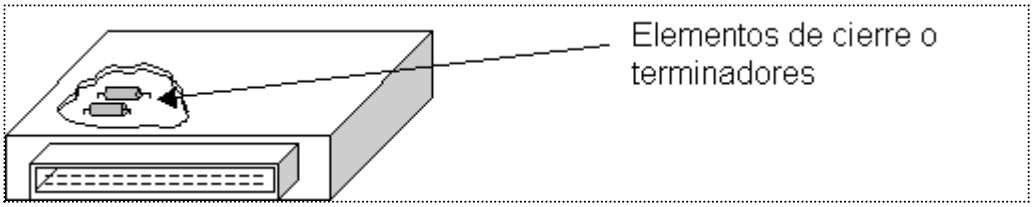

Figura 2.3 Elementos de cierre.

Controlador

Es el elemento encargado de la comunicación entre el microprocesador de la unidad de control y el trasmisor-receptor. Trabaja acondicionando la información que entra y sale entre ambos componentes.

El controlador está situado en la unidad de control, por lo que existen tantas unidades como controladores conectados al sistema. Este elemento trabaja con niveles de tensión muy bajos y es el que determina la velocidad de trasmisión de los mensajes, que será más o menos elevada según el compromiso del sistema. Este elemento también interviene en la necesaria sincronización entre las diferentes unidades de mando para la correcta emisión y recepción de los mensajes. La Figura 2.4 indica en que parte de una unidad de control se encuentra el controlador.

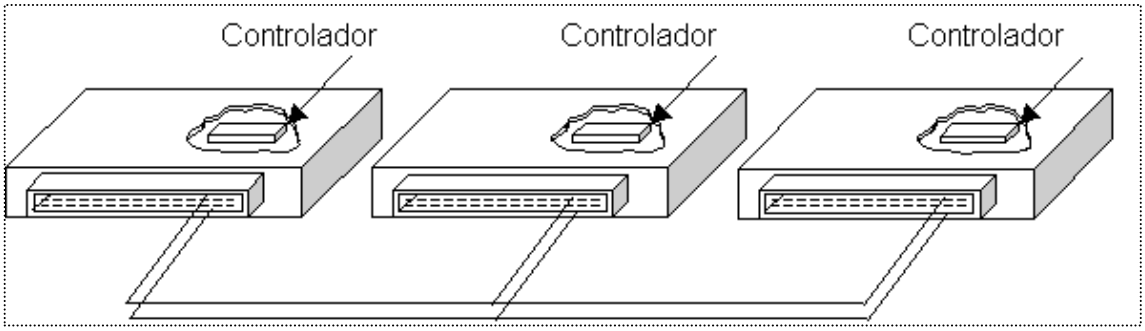

Figura 2.4 Controlador CAN.

Transmisor / Receptor.

El transmisor-receptor es el elemento que tiene la misión de recibir y de trasmitir los datos, además de acondicionar y preparar la información para que pueda ser utilizada por los controladores. Esta preparación consiste en situar los niveles de tensión de forma adecuada, amplificando la señal cuando la información se vuelca en la línea y reduciéndola cuando es recogida de la misma y suministrada al controlador.

El trasmisor-receptor es básicamente un circuito integrado que está situado en cada una de las unidades de control abonadas al sistema, y trabaja con intensidades próximas a 0.5 A y en ningún caso interviene modificando el contenido del mensaje. Funcionalmente, está situado entre los cables que forman la línea CAN-Bus y el controlador. Los elementos ya antes mencionados, son representados en la Figura 2.5.

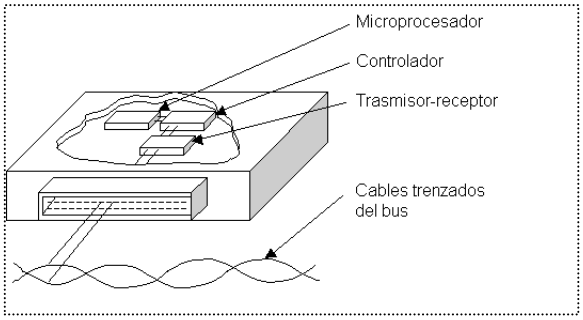

Figura 2.5 Elementos del nodo CAN.

2.1.4 Funcionamiento.

Las unidades de mando que se conectan al sistema CAN-Bus son las que necesitan compartir información, pertenezcan o no a un mismo sistema. En automoción, por ejemplo, están conectadas a una línea las unidades de control del motor, del ABS (Antiblockier system) y del cambio automático, y a otra línea (de menor velocidad) las unidades de control relacionadas con el sistema de confort.

El sistema Can-Bus está orientado hacía el mensaje y no al destinatario. La información en la línea es trasmitida en forma de mensajes estructurados, en la que, una parte del mismo es un identificador que indica la clase de dato que contiene. Todas las unidades de control reciben el mensaje, lo filtran y solo lo emplean las que necesitan dicho dato.

Naturalmente, la totalidad de unidades de control abonadas al sistema son capaces tanto de introducir como de recoger mensajes de la línea. Cuando el bus está libre, cualquier unidad conectada puede empezar a trasmitir un nuevo mensaje.

En el caso de que una o varias unidades pretendan introducir un mensaje al mismo tiempo, lo hará la que tenga una mayor prioridad. Esta prioridad viene indicada por el identificador.

Acceso al bus.

El acceso al bus es orientado a eventos y es puesto aleatoriamente. Si dos nodos desean ocupar el bus simultáneamente, el acceso es implementado con un no destructivo, arbitraje bit a bit, se dice no destructivo cuando el nodo gana el acceso al

bus, posteriormente manda el mensaje, sin que otros nodos intervengan o modifiquen el mensaje. La asignación de prioridad a los mensajes en el identificador es una característica de CAN que lo hace particularmente atractivo para su uso en un entorno de control en tiempo real. Cuanto menor sea el identificador del mensaje binario, el menor tiene mayor prioridad.

En la Figura 2.6 se presenta el proceso de acceso del bus que automáticamente adopta un controlador CAN. Cuando dos nodos intentan transmitir al mismo tiempo el

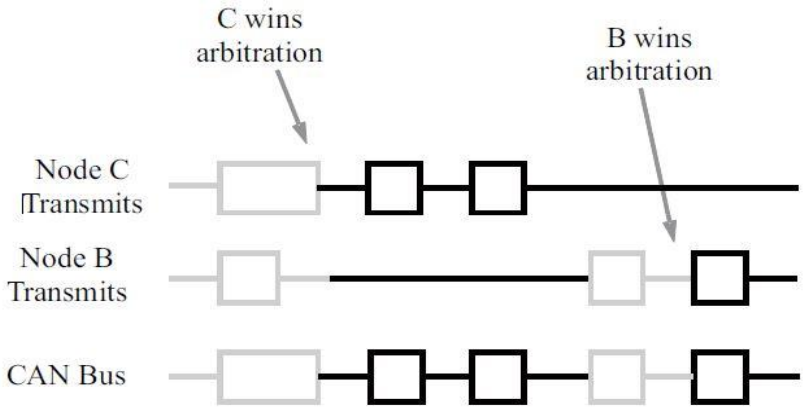

Figura 2.6. Acceso al bus CAN.

nodo de mayor prioridad envía el bit identificador dominante mientras que los otros nodos envían un bit recesivo. Cuando el nodo de mayor prioridad gana la transmisión, los demás nodos esperan a que termine el envío y posteriormente los demás nodos envían.

El sistema CAN-Bus dispone de mecanismos para detectar errores en la trasmisión de mensajes, de forma que todos los receptores realizan un chequeo del mensaje analizando una parte del mismo, llamado campo CRC (Comprobación de Redundancia Cíclica.). Otros mecanismos de control se aplican en las unidades emisoras que monitorizan el nivel del bus, son por ejemplo, la presencia de campos de formato fijo en el mensaje (verificación de la trama), análisis estadísticos por parte de las unidades de mando de sus propios fallos etc.

Estas medidas hacen que las probabilidades de error en la emisión y recepción de mensajes sean muy bajas, por lo que es un sistema extraordinariamente seguro.

El planteamiento del CAN-Bus, como puede deducirse, permite disminuir notablemente el cableado en una red CAN, puesto que si una unidad de mando dispone de una información, como por ejemplo, la temperatura del motor, ésta puede ser utilizada por el resto de unidades de mando sin que sea necesario que cada una de ellas reciba la información de dicho sensor (por ejemplo, en un automóvil).

Otra ventaja es que las funciones pueden ser repartidas entre distintas unidades de mando, lo que incrementa las funciones de las mismas y no presupone un coste adicional excesivo.

2.1.5 La trama.

El mensaje es una sucesión de “0” y “1”, que como se explicaba al principio, están representados por diferentes niveles de tensión en los cables del CAN-Bus y que se denominan “bit”.

El mensaje tiene una serie de campos de diferente tamaño (número de bits) que permiten llevar a cabo el proceso de comunicación entre las unidades de mando según el protocolo definido por Bosch para el CAN-Bus, que facilitan desde identificar a la unidad de mando, así como indicar el principio y el final del mensaje, mostrar los datos, permitir distintos controles etc.

Los mensajes son introducidos en la línea con una latencia que oscila entre los 7 y los 20 milisegundos dependiendo de la velocidad del área y de la unidad de mando que los introduce. La trama de datos estándar es representada en la Figura 2.7, en donde se clasifica cada sección de la trama y demuestra cuando un valor es dominante o recesivo.

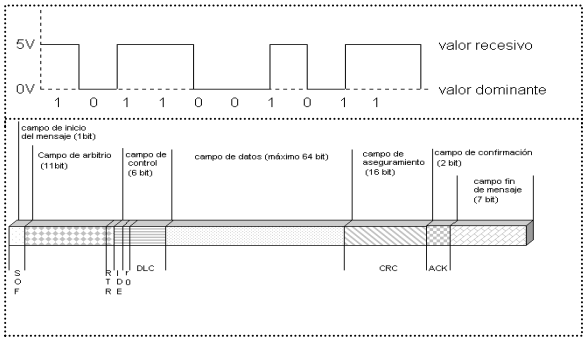

Figura 2.7. La trama y sus partes.

El bus can 2.0A se refiere a un nodo con una trama estándar y el 2.0B a una trama extendida. La diferencia entre una trama estándar y una trama extendida es que la primera tiene 11 bits y la segunda 29 bits. Ambas pueden coexistir eventualmente, y la razón de su presencia es la existencia de dos versiones de CAN.

Puede ocurrir que en determinados mensajes se produzcan largas cadenas de ceros o unos, y que esto provoque una pérdida de sincronización entre unidades de mando. El protocolo CAN resuelve esta situación insertando un bit de diferente polaridad cada cinco bits iguales: cada cinco “0” se inserta un “1” y viceversa. La unidad de mando que utiliza el mensaje, descarta un bit posterior a cinco bits iguales. Estos bits reciben el nombre de “bit stuffing”.

Tabla 2 Bits de la trama

SOF	identificador	RTR	DE	DLC	DATO byte 1	DATO byte 2	CRC	ACK	FN
0	1100010000	0	000	0010	00010110	00000000	0	01	11111

2.1.6 Tipos de tramas en el protocolo CAN.

En [20] explica que el protocolo CAN puede tener los siguientes tipos de tramas:

Trama de datos estándar (Standart Data Frame).

Una trama de datos estándar es generada por un nodo, cuando el nodo desea transmitir datos. El trama comienza con un SOF (Start-of-Frame-estado dominante) para la dura sincronización de todos los nodos.

El SOF es seguido por un campo arbitrario de 12 bits, Los 11 bits de identificación (reflejando el contenido y la prioridad de los mensajes) y el RTR bit (Remote transmission request bit). El siguiente campo es el campo de control, que consiste de 6 bits. El primer bit de este campo es llamado el IDE (Identifier Extension bit) y se encuentra en estado dominante para especificar que el trama es estándar. El siguiente bit es reservado por el protocolo CAN, y está definido como un bit dominante. Los 4 bits restantes del campo de control son el llamado DLC (Data Length Code) y especifica el número de bytes de datos contenidos en el mensaje.

Los datos que se envían en el siguiente campo de datos que es de la longitud definida por el DLC anteriormente (0-8 bits).

El campo CRC (Cyclic Redundancy Check) es usado para detector los posibles errores de transmisión. El campo CRC consiste de una secuencia CRC de 15 bits y un bit delimitador. El mensaje es completado por un campo EOF (End-OF-Frame), que consiste de 7 bits recesivos sin relleno de bits.

El campo final es el campo reconocido. Durante el espacio del bit ACK el nodo transmisor envía un bit recesivo. Cualquier nodo que ha recibido una trama sin errores, reconoce la correcta recepción de la trama mediante el envío de vuelta en un bit dominante (independientemente si el nodo está configurado para aceptar ese mensaje específico o no). El delimitador recesivo conocido completa el espacio conocido y no puede ser sobrescrito por un bit dominante, excepto cuando una trama de error ocurre. La empresa Microchip en [20] detalla de manera gráfica la trama estándar como se muestra en la Figura 2.8.

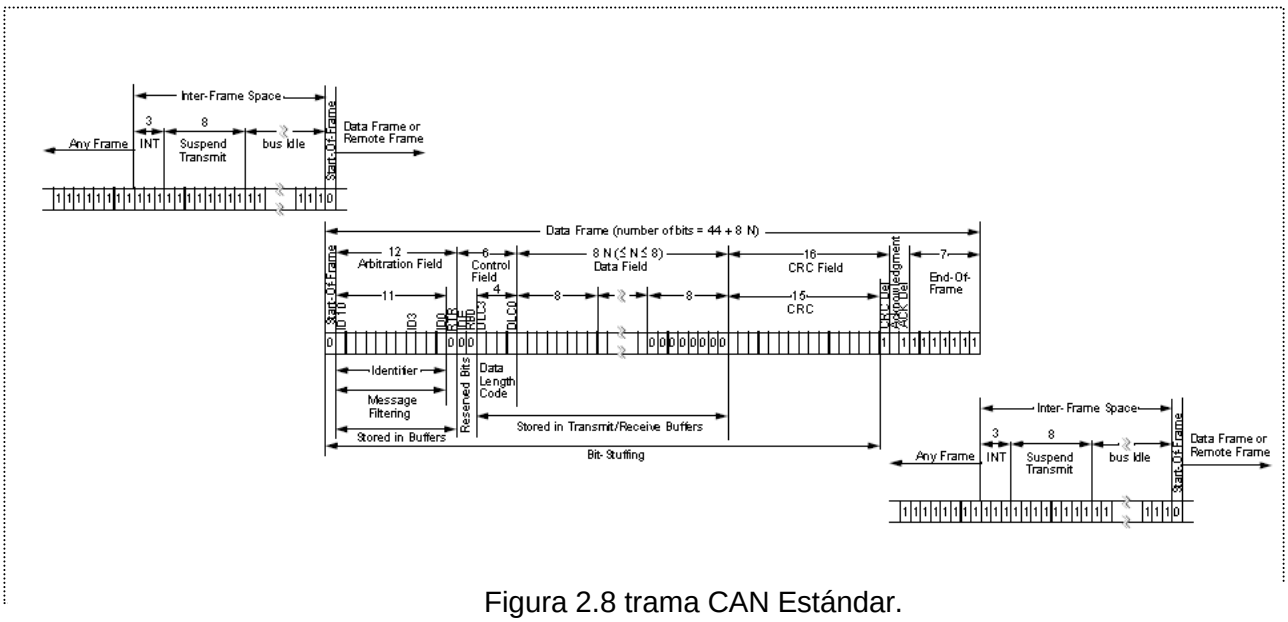

Figura 2.8 trama CAN Estándar.

Trama de datos extensa. (Extended Data Frame).

En las tramas de datos extensas, el SOF (Start-Of-Frame) es seguido por el campo arbitrario de 38 bits. Los primeros 11 bits son los bits más significativos de los 29 bits identificadores (“Base-ID”). Estos 11 bits son seguidos por un SRR (Substitute

Remote Request Bit), el cual es transmitido como recesivo. El SRR es seguido por el bit IDE (Identificación) que es recesivo para denotar que es una trama CAN extenso. Debe tenerse en cuenta de esto, que si el arbitraje sigue sin resolverse después de la transmisión de los primeros 11 bits de la identificación, y uno de los nodos involucrados en el arbitraje está enviando una trama estándar CAN (11 bit), el trama ganará arbitraje por la afirmación de un bit IDE dominante. También, el bit SRR es un trama extenso podría ser recesivo para permitir la afirmación de un bit dominante RTR por un nodo que está enviando una trama CAN remota. Los bits SRR e IDE son seguidos por los 18 bit restantes del identificador (“ID-Extensión”) y un bit dominante RTR.

La porción restante de la trama (campo de datos, campo CRC, campo conocido, EOF e intermisión) se construye en un mismo camino que la trama estándar. Esta trama es visualizada en la Figura 2.9.

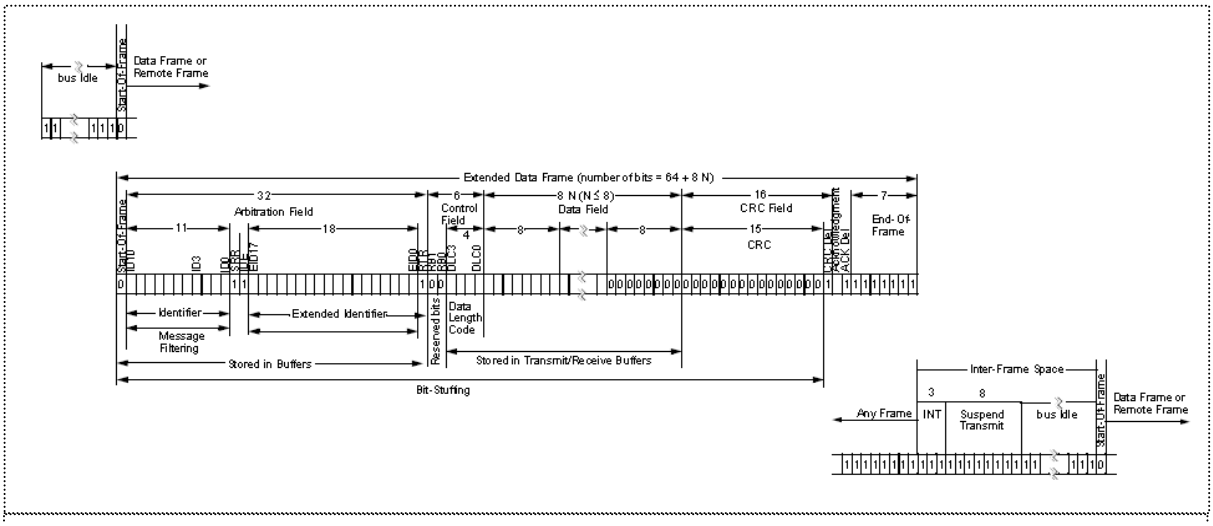

Figura 2.9 Trama CAN extendida.

Para habilitar las tramas extensas y estándar, para ser enviados a través de una red compartida, es necesario dividir el mensaje de 29 bits en secciones de 11 bit (bit más significativos) y 18 bit (menos significativo). Esta división asegura que el bit IDE puede permanecer en la misma posición en ambos tipos de tramas.

El siguiente campo, es el campo de control, consistiendo de 6 bits. Los primero 2 bits del campo son reservados y están en estado dominante. Los 4 bits restantes del campo de control son el DLC (Data Length Code) y especifican el número de bytes de datos.

Trama remoto (Remote Frame).

Una transmisión de datos se realiza de forma autónoma, con el nodo donde se originan los datos (por ejemplo un sensor enviando tramas de datos). Esto es posible no obstante para un nodo destinado a solicitar datos desde la fuente. Para este funcionamiento, el nodo de destino envía una trama remota con un identificador que coincide con el identificador de las tramas de datos requeridas. Luego el nodo de origen, envía una trama de datos como una respuesta a esta solicitud remota.

Hay dos diferencias entre la trama remota y la trama de datos. Primero, el bit RTR está en estado recesivo y segundo no hay campo de datos. En el caso muy improbable de que la trama de datos y la trama remota tengan el mismo identificador al momento de transmitir, la trama de datos gana arbitraje por el bit RTR dominante tras el identificador. En este camino, el nodo que trasmite la trama remota recibe los datos deseados inmediatamente. La trama remota es mostrada en la Figura 2.10.

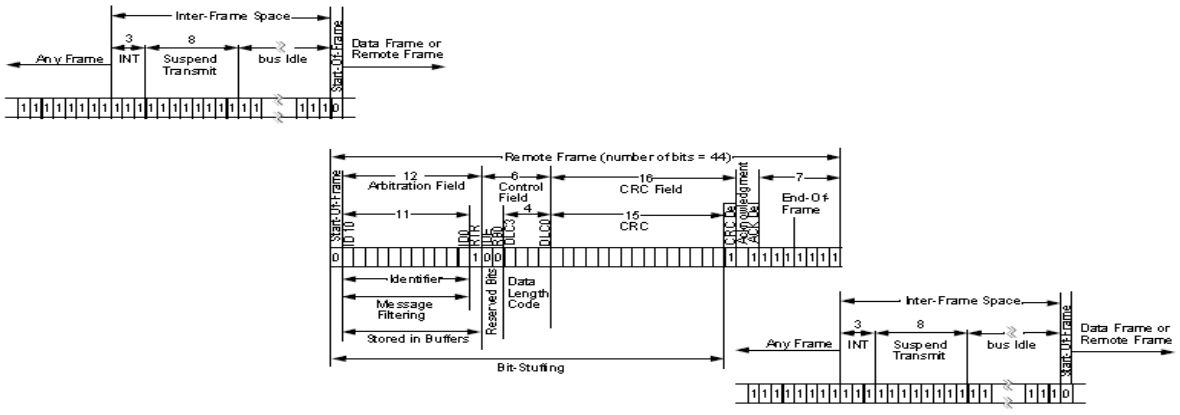

Figura 2.10 Trama remota.

La trama de error.

Una trama de error es generado por algún nodo que detecta un error en el bus. Una trama de error, consiste de 2 campos, un campo de bandera de error seguido por un campo delimitador de error. El delimitador de error consiste de 8 bits recesivos y permite a los nodos reinicializar el bus de comunicaciones limpiándolos después de un error. Hay dos formas de campos de banderas de error. La forma del campo de bandera de error depende del estado del error del nodo que detecta el error.

Si un nodo de error-activo detecta un error de bus, a continuación, el nodo interrumpe la transmisión del mensaje actual mediante la generación de un indicador de error activa. El indicador de error activo (la bandera o flag) está compuesto de 6 bits consecutivos dominantes. Esta secuencia de bit activamente viola la regla de relleno de bits. Todas las demás estaciones reconocen el error relleno de bits resultante y, a su vez generan tramas de error propio, denominado banderas eco de error. El campo de bandera de error por lo tanto consiste de entre 6 y 12 bits consecutivos dominantes (generado por uno o más nodos). El campo delimitador de error completa la trama de error. Después de completar la trama de error, la actividad del bus se mantiene normal y el nodo interrumpido intenta reenviar el mensaje abortado.

Si un nodo de error pasivo detecta un error de bus entonces el nodo transmite un indicador de error pasivo (flag) seguido, nuevamente, por el campo delimitador de error. El indicador de error pasivo, consiste de 6 bits consecutivos recesivos. De esto sigue que, a menos que el error de bus es detectado por el nodo de transmisión u otro error activo recibido que está actualmente transmitiendo, la transmisión de una trama de error por un nodo de error pasivo no afectará algún otro nodo en la red. Si el nodo maestro del bus genera un indicador (flag) de error pasivo, entonces esto podría causar que otros nodos generen tramas de error debido al resultado de la violación de bit de relleno. Después de la transmisión de una trama de error, un nodo de error pasivo debe esperar para 6 bits consecutivos recesivos en el bus antes de reincorporarse a la comunicación del bus. En la figura Figura 2.11 muestra las secciones en la que se divide la trama de error.

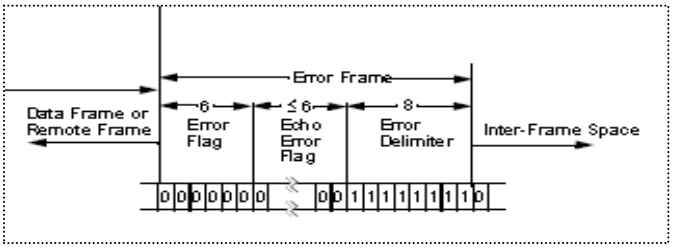

Figura 2.11 Trama de error.

El espacio de inter-trama

El espacio de Inter-trama separa un marco de proceder (de cualquier tipo) de un siguiente dato o trama remoto. El espacio de inter-trama está compuesto de al menos 3 bits recesivos, llamados la intermisión. Este es proporcionado para permitir a los nodos tiempo para el procesamiento interno de los mensajes mediante la recepción de los nodos, antes del comienzo de la siguiente trama de mensaje. Después la intermisión, la línea de bus se mantiene en estado recesivo hasta el inicio de la siguiente transmisión.

Si el nodo transmisor está en estado de error pasivo, se adiciona 8 bits recesivos en el espacio inter-trama antes que algún otro nodo transmita por el bus.

2.1.7 Tipos de implementaciones CAN.

Viendo la estructura del Mailbox del bus, básicamente hay 2 formas de implementación CAN. Estas implementaciones son el Basic CAN y el Full CAN y la diferencia entre las dos, la forma en que llegan los mensajes del bus. Los mensajes son filtrados antes de ser guardados o descartados en el Mailbox. El Mailbox sirve como una interface para el procesamiento que puede acceder al mensaje recibido. Además, las implementaciones actuales varían dependiendo del número de Mailboxes. El filtrado solamente es en el identificador del mensaje recibido. La estructura Basic CAN proporciona una especie de “memoria asociativa” filtrado en el mensaje recibido [1]. Este filtro consiste de un “selector de identificador del mensaje recibido” y un relacionado “mascara de identificación del mensaje recibido”. El identificador parte de cualquier mensaje entrante es entonces comparado con el Id-selector mientras los demás bits son ignorados gracias a la ID-mascara. También el Basic CAN provee de solamente un filtro de hardware asociado, así mismo, el Basic

CAN puede proveer de un buffer de mensaje transmitido, por lo tanto, si se desea transmitir el mensaje solo se le asigna su correspondiente identificador y la información en el buffer de datos. La estructura general del Basic CAN se observa en la Figura 2.13.

El Full CAN podría ser clasificado como un Basic CAN especializado, en donde el ID- mascara es totalmente transparente; por lo tanto no es necesario el registro para la definición de la máscara, como consecuencia, solo un mensaje coincidiría por cada selector identificado recibido. Como consecuencia no se requerirá el identificador post-filtrado en el software del controlador. Por otra parte, esta técnica requiere múltiples buffers para ser implementados en el bus CAN. Una desventaja es que cada nodo solo podrá tener 16 Mailbox por cada nodo. La implementación Full CAN lo podemos ver en la Figura 2.12.

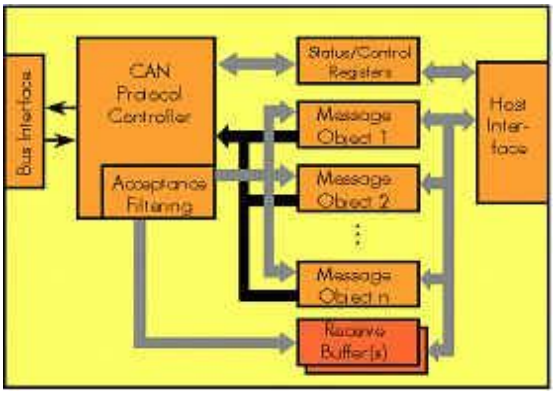

Figura 2.12 Implementación Full CAN

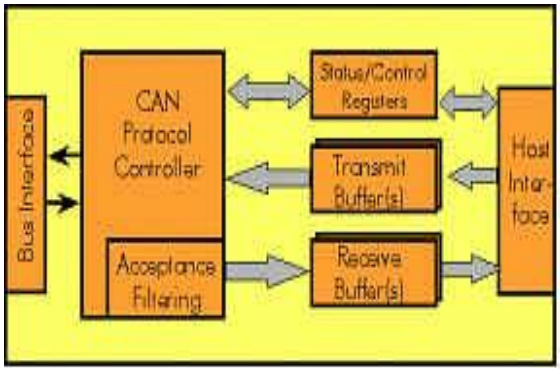

Figura 2.13 Implementación Basic CAN.

CAN de alta velocidad.

La ISO11898-2 es la especificación utilizada para el bus CAN de media y alta velocidad. El bus consiste de dos líneas CAN_H y CAN_L y una señal común de tierra. También especifica que el bus debe tener una resistencia de 120 ohm en cada terminación del bus, para minimizar las reflexiones y picos de la forma de onda. Los estados lógicos de los bits son determinados por un voltaje diferencial entre las dos líneas. En la Figura 2.14 se visualiza la implementación del bus CAN con la ISO 11898-2.

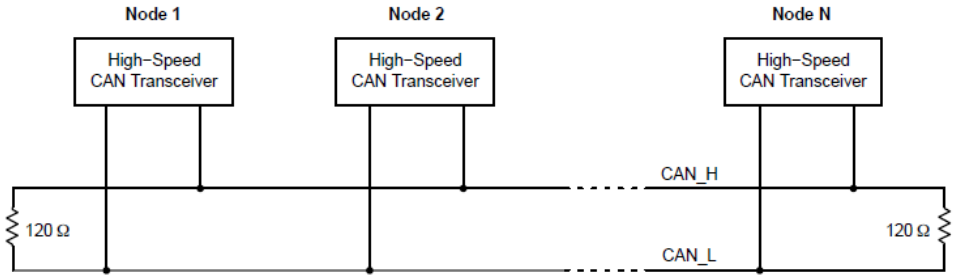

Figura 2.14 Implementación del bus con la ISO 11898-2.

2.2 Teoría de control.

La teoría de control, es un tema muy estudiado en donde existe una amplia literatura y se han desarrollado técnicas para el diseño de controladores(o compensadores), ya sea utilizando la teoría de control clásica o moderna, el objetivo es poder diseñar el compensador ideal de acuerdo al sistema que se desea controlar.

2.2.1 Métodos de identificación de sistemas.

Existen diferentes maneras de identificar un sistema para obtener la función de transferencia. Dichos métodos se pueden clasificar de dos maneras [21]:

Dependiendo de modelo obtenido:

Métodos no paramétricos: Estos Métodos hacen el uso de técnicas como: Análisis de la respuesta transitoria, análisis de la respuesta en frecuencia, análisis de la correlación, análisis espectral, análisis de Fourier, entre otros.

Métodos paramétricos: Estos métodos requieren la elección de una posible estructura del modelo, de un criterio de ajuste de parámetros, y por ultimo dela estimación de parámetros que mejor se ajustan el modelo a los datos muestreados.

Dependiendo de la aplicación:
Métodos de identificación off-line(a posteriori):
Es utilizado en aquellas aplicaciones en que no se requiera un ajuste continuado del modelo. En estos casos, se realiza la identificación previa a la planta, considerándose que la validez de los parámetros obtenidos no se verá alterada con el paso del tiempo.

Métodos de identificación on-line (recursiva): Se realiza la identificación en los que los parámetros se van actualizando continuamente a partir de los nuevos datos de entrada-salida obtenidos durante la evolución del proceso. Esta es muy utilizado en el control adaptativo.

2.2.2 Identificación paramétrica del sistema.

Este método de identificación pretende obtener un modelo matemático del proceso que se comporte de una manera lo más aproximada posible al sistema real. La manera más usual de identificar a un sistema es aplicar en su entrada una excitación conocida, obtener la respuesta y después tratar de ajustar un modelo conocido a las particularidades del sistema sujeto a la identificación [22].
Las señales más usadas como excitación al hacer la identificación son:

- Señal escalón, también conocido como “identificación en el tiempo”
- Señal senoidal de amplitud constante y frecuencia variable, también conocido “identificación en frecuencia”.

Identificación en el tiempo.
Para realizar la identificación en el tiempo, es necesario excitar el sistema por medio de una función escalón, unitaria de preferencia, y conocer la respuesta a

esa excitación. Con la respuesta obtenida se realiza una gráfica C(t) vs t adecuadamente escalada para saber la reacción del sistema a la excitación. Una vez obtenida la gráfica se clasifica la respuesta de acuerdo a una lista de respuestas establecidas. Esto sirve para saber de qué orden es el sistema a diseñar.

Respuestas sobre amortiguadas.

Este tipo de respuesta se puede clasificar de varias maneras:

1º orden puro

La respuesta típica de estos sistemas no presenta sobreoscilación, esto quiere decir que nunca llegan al valor exacto de la consigna y por lo tanto, son sistemas relativamente lentos. Esto se puede apreciar en la Figura 2.15

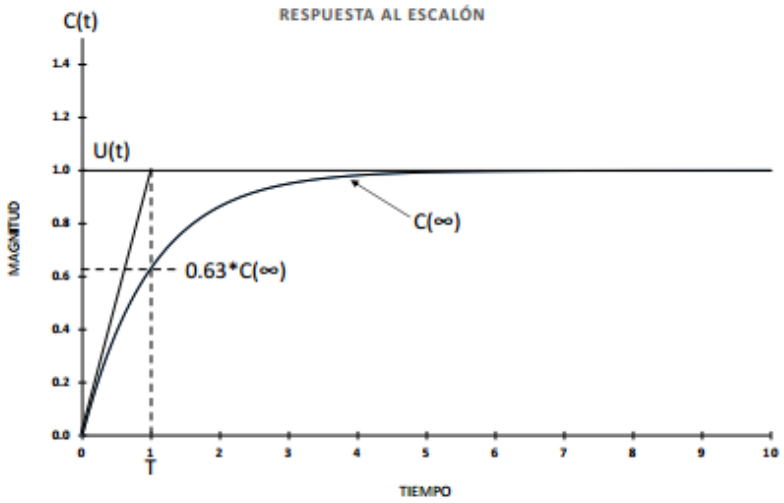

Figura 2.15 Respuesta de primer orden puro.

1º orden con retardo.

La respuesta típica de este tipo de sistemas, presenta la misma configuración que un sistema de 1º orden puro, en el cual la respuesta presenta un desfase o retardo respecto a la señal de entrada. Lo anterior se puede apreciar en la Figura 2.16.

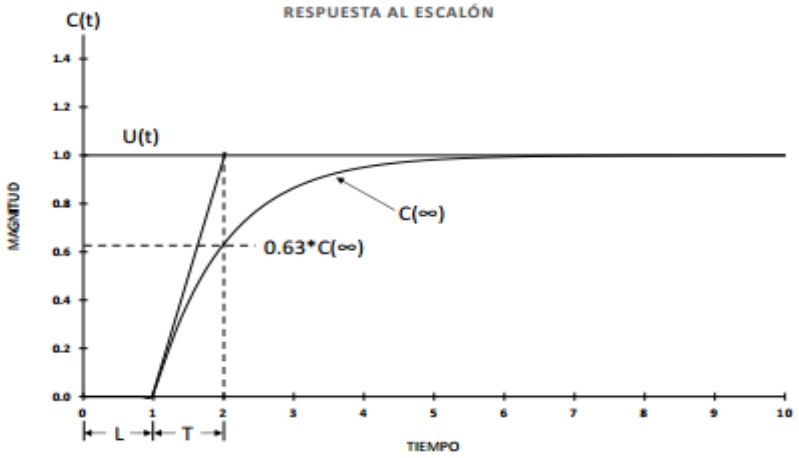

Figura 2.16 Respuesta de 1° orden con retardo.

Polos reales múltiples.

La respuesta de este tipo de sistemas varía según la cantidad de polos existentes, conforme aumenta el número de polos la respuesta es más lenta, teniendo al inicio un arranque con mayor suavidad. Esto se puede apreciar en la Figura 2.17.

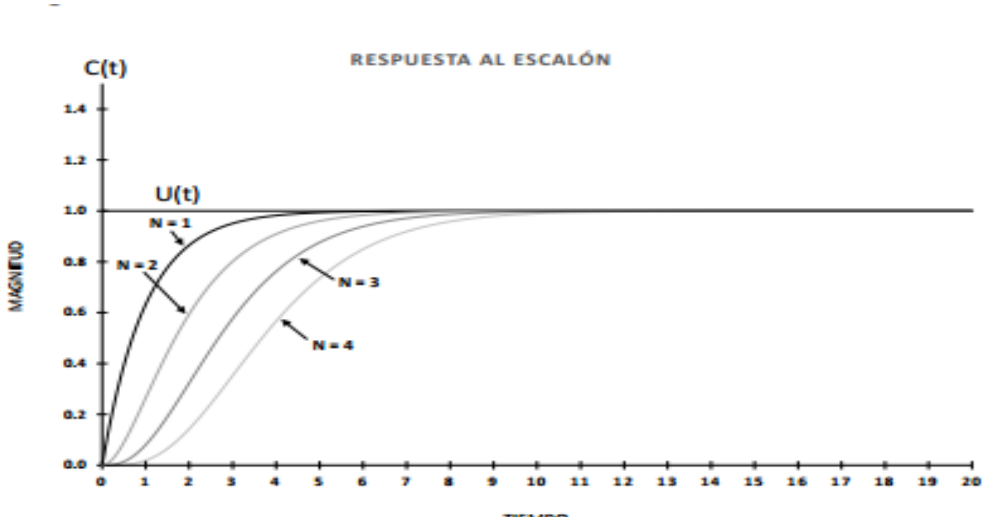

Figura 2.17. Respuesta polos reales múltiples.

Polos reales distintos.

Si la respuesta es sobreamortiguada, y no se ajusta a ningún sistema visto hasta ahora y $\frac{T_u}{T_a}$ < 0.104 (n = 2), se puede ajustar con polos reales distintos. La respuesta típica de una función de transferencia de polos reales distintos se puede apreciar en la Figura 2.18

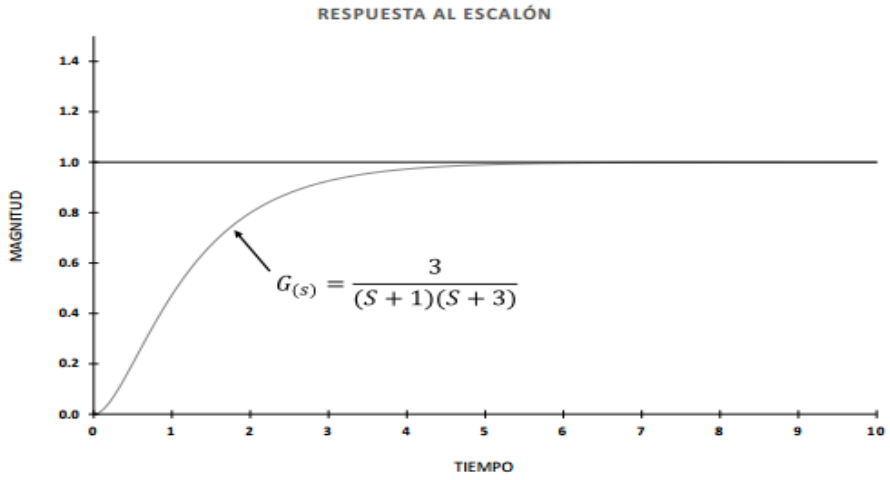

Figura 2.18 Respuesta polos reales distintos.

Respuestas subamortiguadas.

Este tipo de respuestas presentan sobreoscilacion y un periodo transitorio con oscilación, y se deben a sistemas con polos complejos conjugados.

Sistemas estándar de segundo orden.

La mayoría de los sistemas industriales se comportan como un sistema de este tipo, en el cual posteriormente el control pretende limitar parámetros como la sobreoscilacion, tiempo de establecimiento y error en régimen permanente. La respuesta típica de una función de transferencia de polos complejos se puede apreciar en la Figura 2.19.

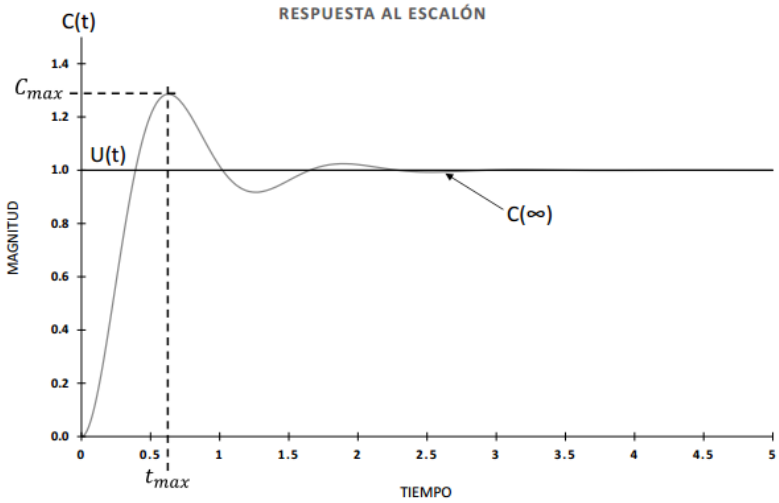

Figura 2.19 Sistema estándar de segundo orden.

Sistemas estándar de segundo orden con retardo.

La respuesta típica de una función de transferencia de polos complejos con un retardo se puede apreciar en la Figura 2.20.

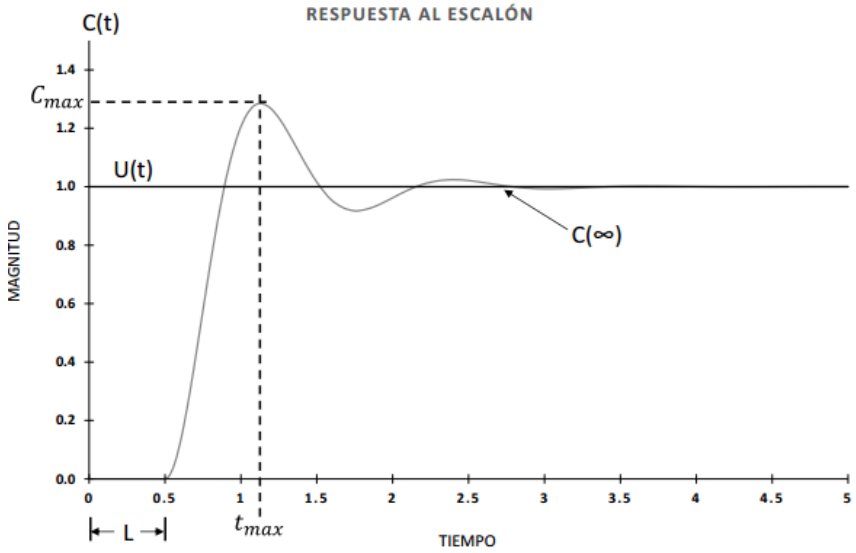

Figura 2.20 Sistema estándar de segundo orden con retardo.

Sistemas inestables.

Un tipo de sistema inestable es un sistema de 1º orden con integrador, cuya función de transferencia es la siguiente.

$$G(s) = \frac{K}{S(T_1 S + 1)}$$

Un criterio simple para poder obtener el parámetro que caracteriza al sistema, es poner en el origen un polo, y proponer una constante de tiempo T_1, hasta obtener la respuesta real. La función de transferencia G(S), debe estar en lazo cerrado.

2.3 Comportamiento del hardware.

2.3.1 PSoC.

El PSoC es un sistema embebido Programable System On Chip, desarrollado por la empresa CYPRESS. Este chip integra puertos, funciones analógicas y digitales programables. También integra memoria y un microcontrolador en un chip.

Dentro de la gama de microcontroladores CYPRESS, el PSoC se clasifica en cuatro "familias": el PSoC 1, PSoC 3, PSoC 4 y PSoC 5. A continuación se presentara las características más importantes del PSoC 5:

El PSoC 5 es un microcontrolador que cuenta con una arquitectura innovadora que permite tener una resolución analógica de 20 bits, una lógica PLD (Programmable Logic Device) programable. Y una Unidad Central de Procesamiento (CPU) ARM Córtex M3 de 32 bits a 67 Mhz. La CPU tiene las siguientes características:

- Arquitectura Harvard.
- 83 MIPS.
- Arquitectura ARM v7 mejorada.
- Conjunto de instrucciones Thumb2.
- 16 y 32 bit de instrucciones (no modificable).
- ALU de 32 bits.

Subsistema CPU del PSoC 5 tiene las siguientes características:

- Memoria flash con ECC (Error-Correcting Code).
- EEPROM.
- 24 canales de acceso a la memoria directa
- Acceso a todos los puertos analógicos y digitales.
- CPU y DMA acceden simultáneamente a bloques independientes de SRAM.
- Cuenta con el estándar industrial JTAG-SWD(Serial Wide Debug)
- Cuenta con múltiples fuentes de reloj.
- Salida PLL.

En cuanto a comunicaciones el PSoC 5 Ofrece:

- Comunicación USB Full Speed.
- Basic y Full CAN 2.0
- I2C maestro o esclavo.
- CapSense.
- Comunicación LIN.
- Comunicación MDIO,SMBUS,I2S.
- Comunicación USB, UART.

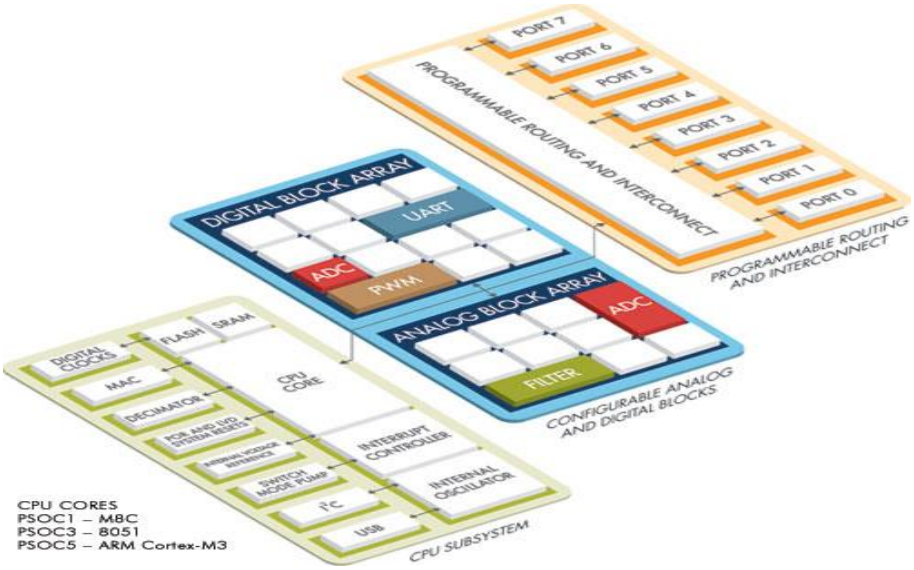

Figura 2.21 Arquitectura PSoC.

Para poder programar este microcontrolador, la empresa Cypress ofrece la herramienta PSoC Creator, que es una plataforma gratuita en donde se puede escribir código en lenguaje C o ensamblador.

2.3.2 Transceptor MCP2551.

El integrado MCP2551 es un transceptor de alta velocidad CAN desarrollado por la empresa MICROCHIP, las características de este chip son:

- Soporta hasta 1Mbps de velocidad de transferencia.
- Implementa la ISO-11898.
- Detección de fallos a tierra (permanente dominante).
- Soporta hasta 112 nodos conectados.
- Protección a corto circuito.
- 3 modos de operación: High Speed, Slope-Control y Stanby.

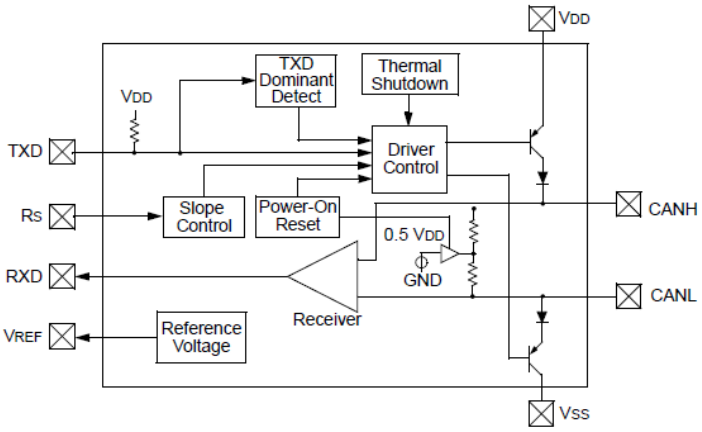

Figura 2.22 Transceptor MCP2551.

Modos de operación.

El transceptor MCP2551 tiene tres modos de operación, high speed, Slope-Control y Standby. Para configurar dichos modos, se tiene que configurar el pin Rs del chip.

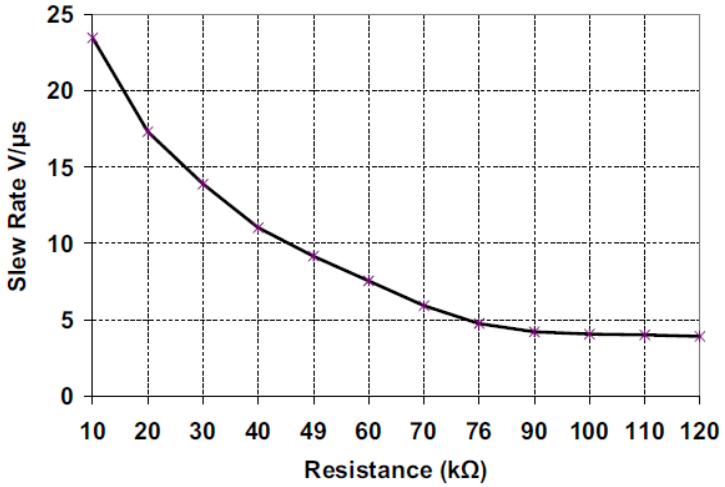

Figura 2.23 Slew rate de MCP2551.

Para configurar el transceptor en modo high Speed, el pin Rs se conecta a Vss. Para configurarlo en modo Slope-Control, el pin Rs se le conecta una resistencia. Dependiendo del valor de la resistencia el slew rate de la señal cambiara, dicho comportamiento se visualiza en la Figura 2.23. Por último, para configurar el modo Stanby, el pin Rs debe ser conectado en un nivel alto.

2.3.2 Sensor de distancia óptico.

El Sharp 2Y0A21 (Figura 2.24) es un sensor medidor de distancias por infrarrojos que indica mediante una salida analógica la distancia medida. La tensión de salida varía de forma no lineal cuando se detecta un objeto en una distancia entre 10 y 80 cm. Normalmente se conecta esta salida a la entrada de un convertidor analógico digital el cual convierte la distancia en un número que puede ser usado por el microprocesador. La salida también puede usada directamente en un circuito analógico. Hay que tener en cuenta que la salida no es lineal. El sensor utiliza solo una línea de salida para comunicarse con el procesador principal.

Figura 2.24 Sensor Sharp 2Y0A21

Este dispositivo emplea el método de triangulación utilizando un pequeño Sensor Detector de Posición (PSD) lineal para determinar la distancia o la presencia de los objetos dentro de su campo de visión.

Su modo de funcionamiento consiste en la emisión de un pulso de luz infrarroja, que se transmite a través de su campo de visión que se refleja contra un objeto. Si no encuentra ningún obstáculo, el haz de luz no refleja y en la lectura que se hace indica que no hay ningún obstáculo. En el caso de encontrar un obstáculo el haz de luz infrarroja se reflecta y crea un triángulo formado por el emisor, el punto de reflexión (el obstáculo) y el detector.

La información de la distancia se extrae midiendo el ángulo recibido. Si el ángulo es grande, entonces el objeto está cerca (el triángulo es ancho). Si el ángulo es

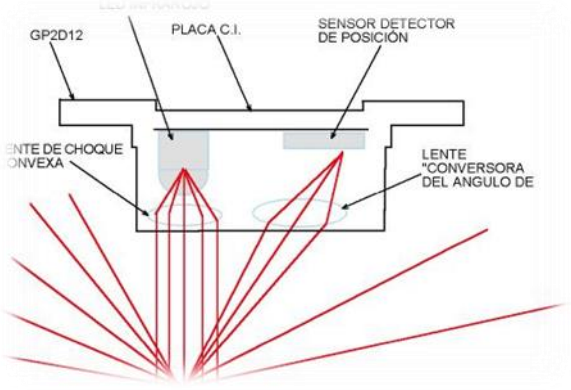

Figura 2.25 Triangulación del sensor Sharp.

pequeño significa que está lejos (el triángulo es largo y, por tanto, delgado).

Observemos ahora como se lleva a cabo la triangulación (Figura 2.25) en el sensor. El LED infrarrojo emite el haz de luz a través de una pequeña lente convergente que hace que el haz emisor llegue de forma paralela al objeto. Cuando la luz choca con un obstáculo, una cierta cantidad de luz se refleja, si el obstáculo fuera un espejo perfecto, todos los rayos del haz de luz pasarían y sería imposible medir la distancia. Sin embargo, casi todas las sustancias tienen un grado bastante grande de rugosidad de la superficie que produce una dispersión hemisférica de la luz (reflexión no teórica). Alguno de estos haces de ésta luz rebota hacia el sensor que es recibida por la lente.

La lente receptora también es una lente convexa, pero ahora sirve para un propósito diferente, actúa para convertir el ángulo de posición. Si un objeto se pone en el plano focal de una lente convexa y los otros rayos de luz paralelos en otro lado, el rayo que pasa por el centro de la lente atraviesa inalterado o marca el lugar focal. Los rayos restantes también enfocan a este punto.

En el plano focal hay un Sensor Detector de Posición (PSD). Éste dispositivo semiconductor entrega una salida cuya intensidad es proporcional a la posición respecto al centro (eficaz) de la luz que incide en él. El rendimiento del PSD en la salida es proporcional a la posición del punto focal. Esta señal analógica tratada es la que se obtiene a la salida del sensor.

2.3.2 Sensor óptico opto-interruptor.

El optoswitch es un sensor ampliamente utilizado para saber si existe la presencia de un objeto. El sensor está conformado por un LED, un fototransistor y una estructura que le permite tanto al LED y al fototransistor estar posicionados paralelamente.

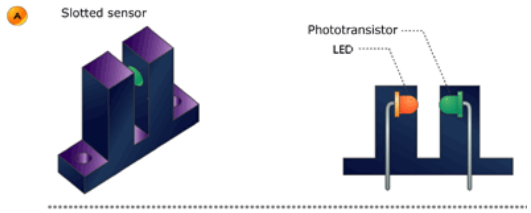

Figura 2.26 Opto-Switch.

El sensor al polarizar directamente al LED, enviará a la “base” del fototransistor fotones que lo polarizará para que permita la salida de un voltaje en el transistor (Figura 2.27). Prácticamente este tipo de sensor es utilizado para la detección de objetos para realizar la función de interruptor o para poder calcular la velocidad en los motores.

Figura 2.27 Funcionamiento del Opto-Switch.

Capítulo 3 Desarrollo.

En este capítulo se presentan los diagramas, esquemáticos y cálculos realizados para la implementación de los sistemas de control planteados en el capítulo 1, también se presenta el software desarrollado para la monitorización de la red CAN.

3.1 Hardware para el PSOC5LP.

Se estableció la tecnología PSoC 5 debido a las características expuestas en el capítulo 2. Los integrados de la familia PSoC 5 son numerosos, cada uno con recursos diferentes, también, de diversos encapsulados. Todos los integrados de esta familia son de montaje superficial, por lo que dificulta la implementación de la misma o los costos de fabricación son elevados. Por ello, La empresa CYPRESS tiene disponible el kit CY8CKIT-059, la cual, tiene implementado un CY8C5888LTI-LP097 de la familia PSoC 5LP, a un costo realmente bajo (10 dólares), dicho kit, permite el uso de los diferentes recursos que ofrece el PSoC 5, también su facilidad

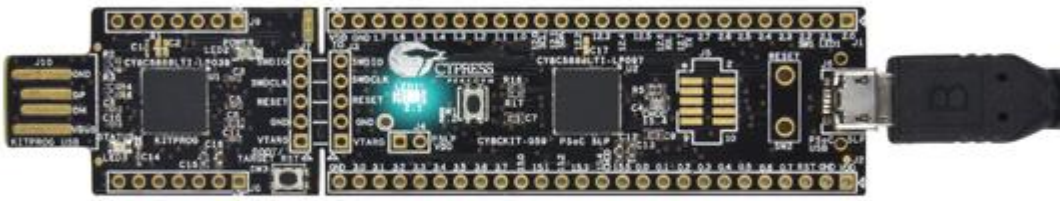

Figura 3.1 CY8CKIT-059

de implementación hace de este un kit muy atractivo.

Otra de las ventajas del CY8KIT-059, es la integración de una tarjeta programadora y depuradora con conexión al puerto USB, que puede desprenderse del kit para dejar solo al microcontrolador.

De acuerdo a los objetivos del proyecto (capitulo 1), se utilizará la comunicación CAN. El CY8C5888LTI-LP097 solo cuenta con un módulo CAN configurable en Basic o Full, pero no cuenta con el transceptor, por lo tanto, se necesita hardware externo

para poder implementar la comunicación de manera exitosa. Esto conlleva a la elaboración de una Shield, en donde se integrará el transceptor correspondiente.

3.1.2 Elaboración de la PCB con el MCP2551.

La elaboración de la PCB (Shield) debe satisfacer los siguientes requerimientos:

- Incorporar el transceptor MCP2551.
- Habilitar o deshabilitar resistores en el bus.
- Permita energizar al kit tanto con una fuente externa o directamente del puerto microUSB.

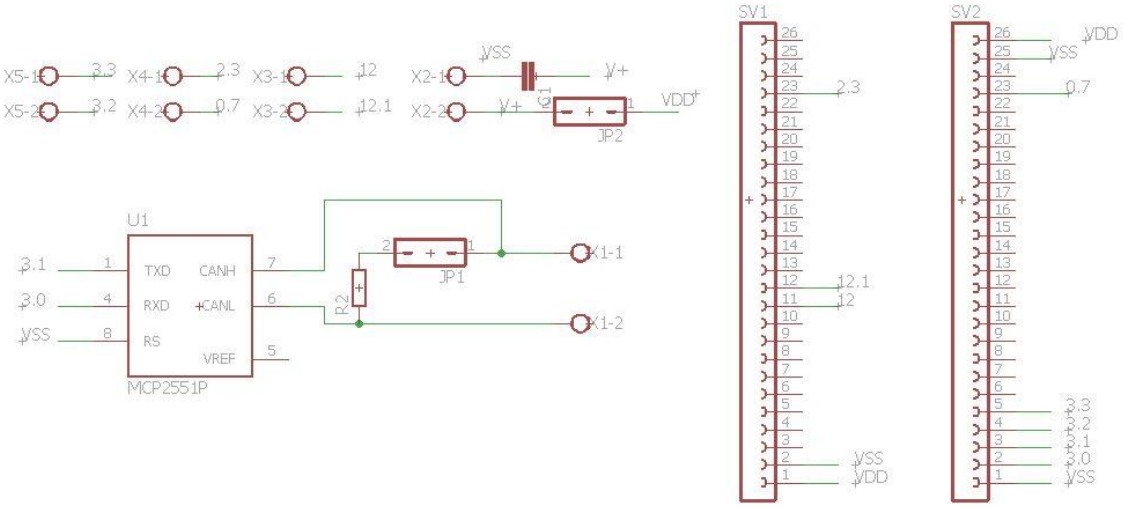

Figura 3.2 esquemático PCB

En la Figura 3.2 se presenta las conexiones para el transceptor MCP2551 dentro de las cuales el jumper JP1 servirá para habilitar o deshabilitar el resistor R2 del esquemático. De la misma forma el jumper JP2 servirá para habilitar la energización por una fuente externa o mediante el cable microUSB. Con dichas conexiones, se procedió a diseñar la PCB (Figura 3.3), donde se buscó colocar los elementos que permitan su uso de manera fácil y práctica.

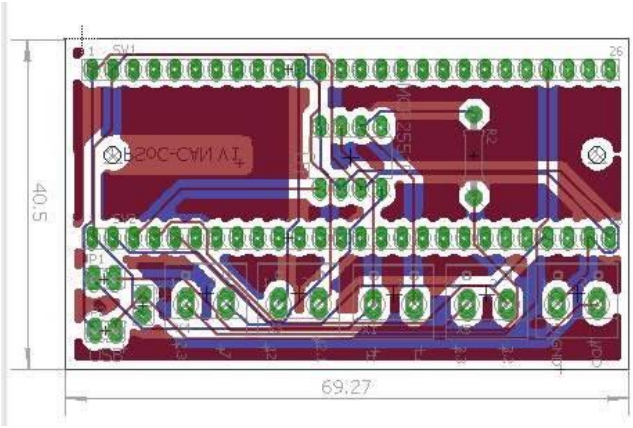

Figura 3.3 PCB MCP2551

3.2 Definición de la red CAN.

El objetivo es crear una red Full CAN con 4 nodos (Figura 3.4), cada nodo contara con una ID, donde el nodo 1 será de alta prioridad. A continuación se describe la tarea de cada nodo:

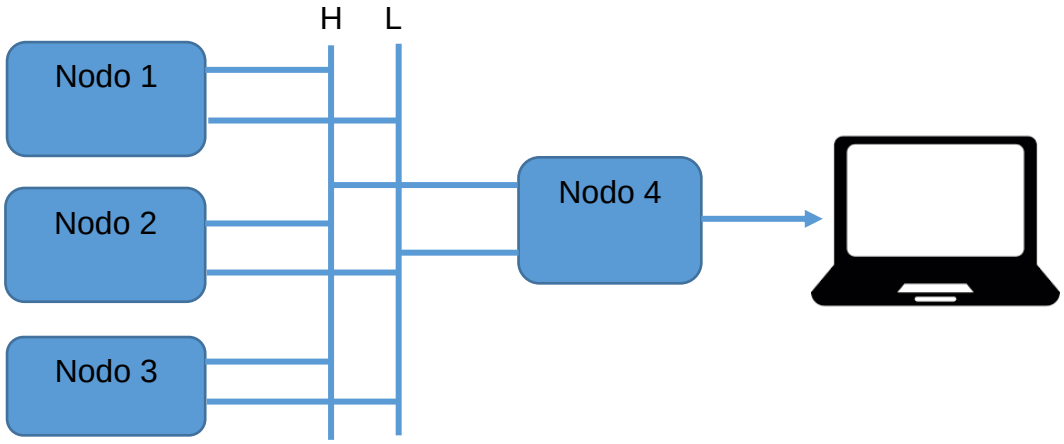

Figura 3.4 Red CAN.

Nodo 1: EL nodo enviará y recibirá datos del sistema 1 en el bus con una implementación full CAN. Este nodo será de alta prioridad.

Nodo 2: Este nodo enviará y recibirá datos del bus con una implementación Full CAN. Este nodo manipulará los sensores o actuadores del sistema 2.

Nodo 3: Este nodo enviará y recibirá datos del bus con una implementación full CAN. Será el responsable para el procesamiento de los controles para los sistemas 1 y 2.

Nodo 4: Este nodo recibirá y enviará datos a bus CAN y enviara datos a la computadora a través de una comunicación USB-HID para la creación de una interfaz, que monitoreará la red.

3.3 Definición de los sistemas.

Los sistemas propuestos, servirán para crear tránsito de datos en la red, también para utilizar las técnicas de identificación de sistemas a partir de datos experimentales. Es importante recalcar que el nodo 3 será el responsable del procesamiento de los compensadores de los sistemas propuestos, esto es con el fin de crear más tránsito de datos en la red CAN. Por lo tanto, los nodos que corresponden a su respectivo sistema, solo enviarán datos de la lectura de sus sensores y recibirán los valores calculados (del nodo 3) para controlar sus actuadores.

Sistema 1. Motor DC con clutch magnético.

El sistema 1 estará conformado de un motor de corriente continua de 24 vdc, que tendrá un clutch magnético que lo frene. Se diseñará el control apropiado, parque que a determinada perturbación (frenado) del sistema, este mantenga su velocidad. Los parámetros a satisfacer en el control será que el sistema responda a no más de 1.5 segundos con menos de 5% de sobretiro y un error en estado estacionario de 1%. El diagrama a bloques del sistema 1 se visualiza en la Figura 3.5

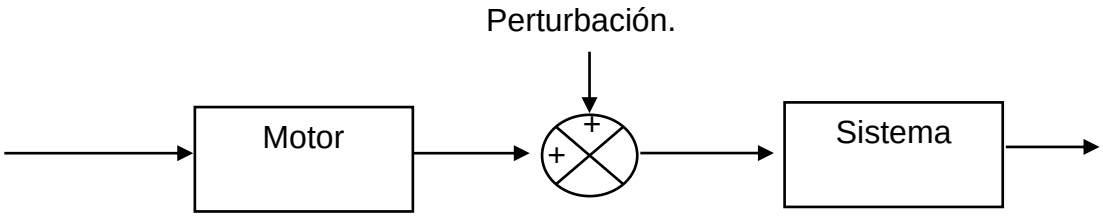

Figura 3.5 Diagrama a bloques del sistema 1.

Sistema 2. Levitador Neumático.

El sistema 2 estará representado por un levitador neumático. Esta comprende de hacer “levitar” un disco dentro de una superficie cilíndrica hueca (tubo) en una distancia deseada. El objetivo de este sistema es diseñar un control para que el disco mantenga la distancia establecida cuando se presenta una perturbación. El control que se diseñará debe cumplir lo siguiente: debe responder en dos segundos con un error en estado estacionario menor al 2%, el sobretiro debe ser menos al 5%.

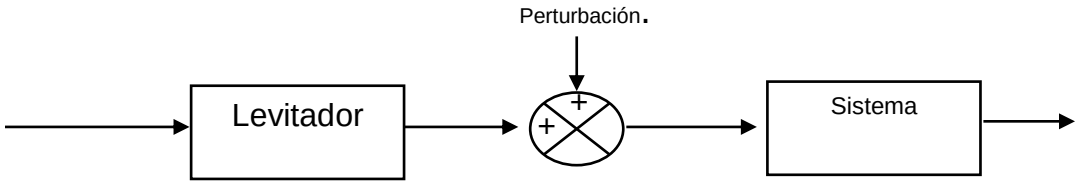

Figura 3.6 Diagrama a bloques del sistema 2.

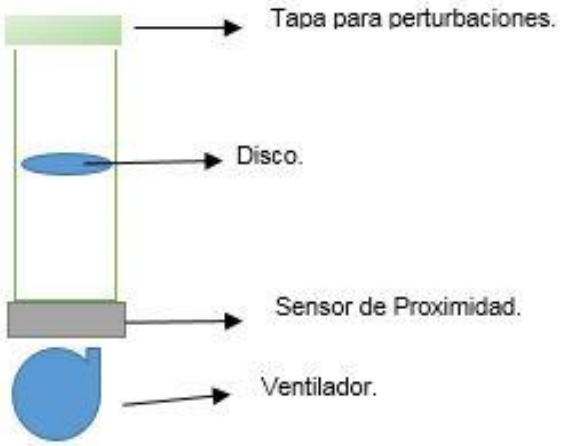

Figura 3.7 Estructura del levitador.

En la Figura 3.7 se presenta la estructura del levitador, la cual va a estar conformada de una tapa para generar perturbaciones, un disco que se elevara bajo la acción de un ventilador. La distancia va a ser medid por medio de un sensor de proximidad.

3.3 La interfaz CAN.

De acuerdo a los objetivos, se creará una interfaz para el monitoreo de la red, así como para saber el comportamiento de los sistemas. Esta será una herramienta muy importante ya que con ella obtendremos las gráficas para el diseño de los controles. Para enviar lo datos a la computadora se utilizará el protocolo de comunicación USB-HID, debido al fácil instalación en cualquier computadora, sin necesidad de algún archivo extra. El PSoC 5 tiene la flexibilidad de tener varios tipos de comunicación USB, dentro de los cuales está el HID. Para la realización de esta tarea, se explicara en 2 secciones: la primera sección explicará la realización del firmware que permitirá funcionar el hardware. Y la segunda sección, la programación de la interfaz.

La realización del firmware se utilizó, el software PSoC Creator, que es la IDE para programar en el PSoC 5. En la Figura 3.8 se muestra los elementos utilizados para la programación del firmware.

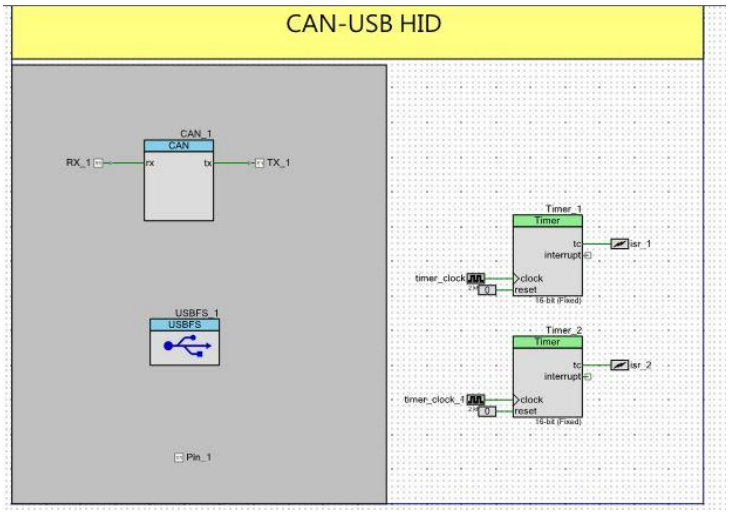

Figura 3.8 Firmware Psoc creator.

En la Figura 3.8 se muestra la utilización de 2 temporizadores, que servirán para actualizar variables utilizadas en los módulos CAN y USBFS. Por otra parte, el modulo CAN tiene que ser configurado para tener una implementación Full CAN, como se muestra en la Figura 3.9, también, se habilitan las opciones de configuración a un mailbox como Full en el PSoC Creator. Además, se habilitan 4

mailbox que serán usados para la recepción de datos de los diferentes nodos. Siguiendo con la configuración del módulo, en la Figura 3.10 se muestra los parámetros para alcanzar una velocidad de transmisión de 1Mbps.

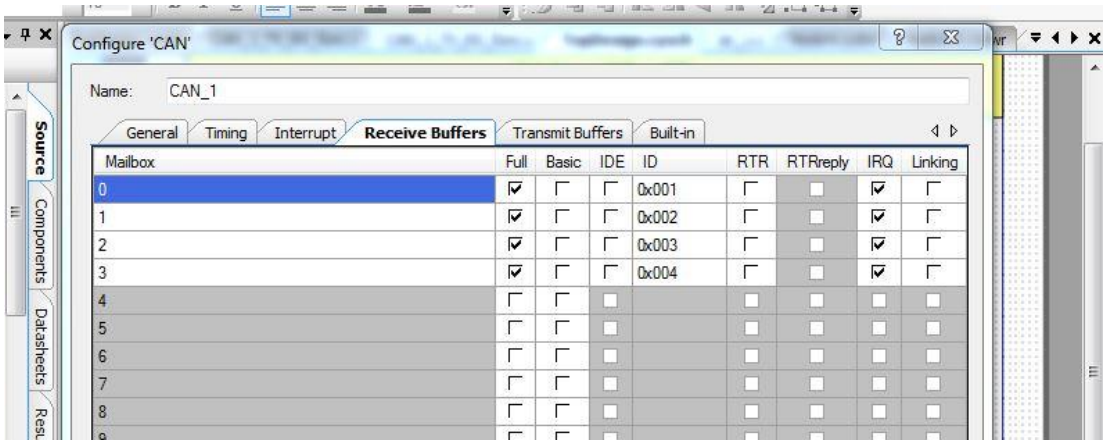

Figura 3.9 Configuración Full CAN.

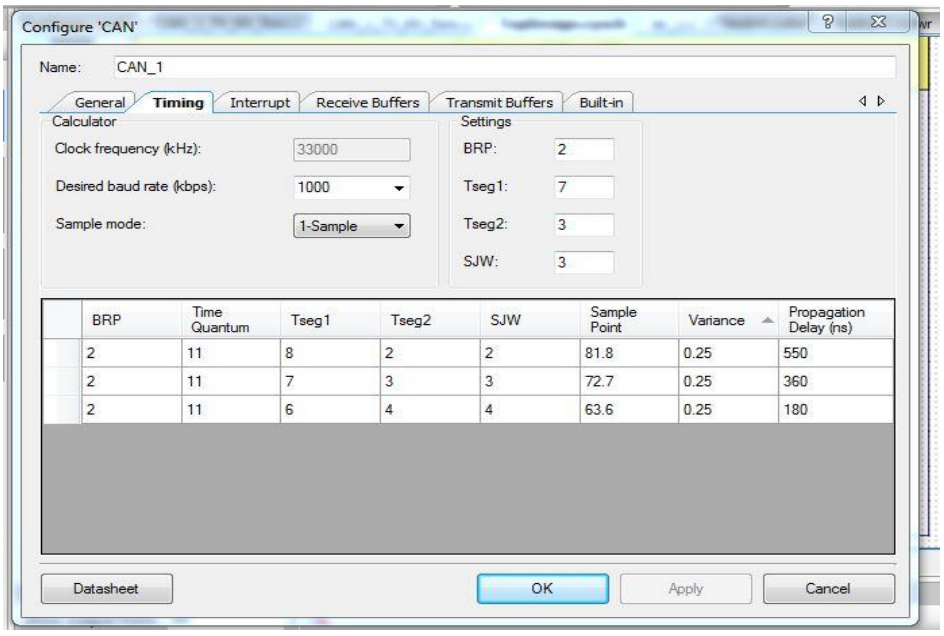

Figura 3.10 Configuración 1Mbps

Los valores establecidos son una BRP =2 Tseg1 = 7 Tseg2 = 3 SJW = 3. Dichos valores establecen una varianza de 0.25 y un retardo de propagación de 350ns.Por otra parte para establecer una comunicación USB-HID, se tiene que crear un descriptor, donde se declaran los parámetros para establecer una comunicación de tipo HID con la computadora. En la Figura 3.11 se presenta dicha configuración, dentro de la cual permite una comunicación de entrada y salida de 8 bytes.

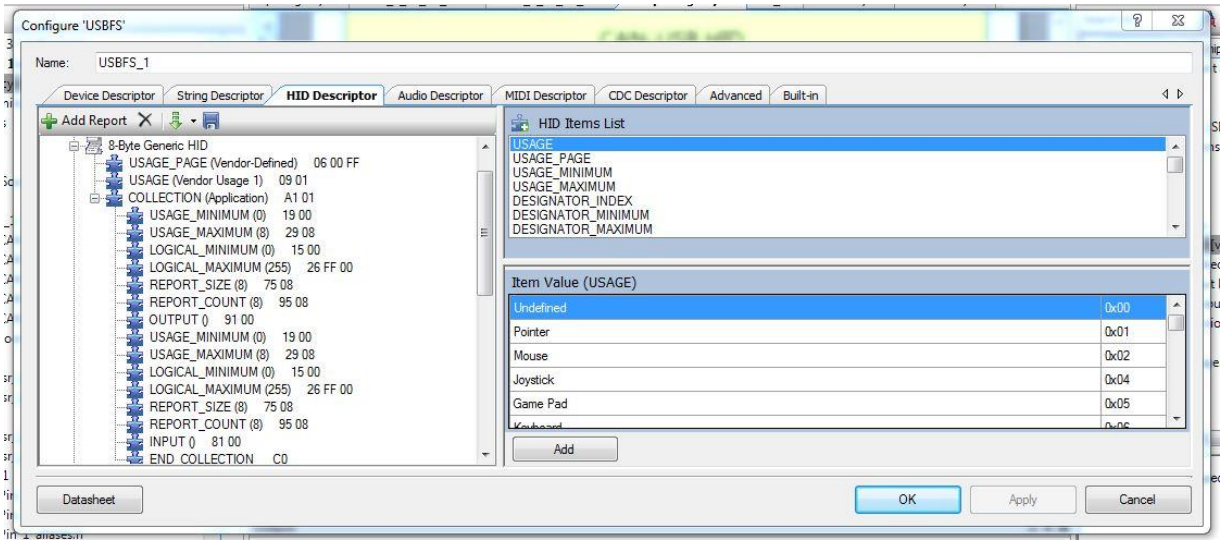

Figura 3.11. Descriptor HID.

Una vez configurado tanto los parámetros para los módulos CAN y USBFS, se realiza la programación necesaria en el microcontrolador para que pueda recibir los valores del bus y enviarlos a la computadora. A continuación se presenta un diagrama de flujo para explicar la lógica de la programación en el nodo 4 de la red CAN.

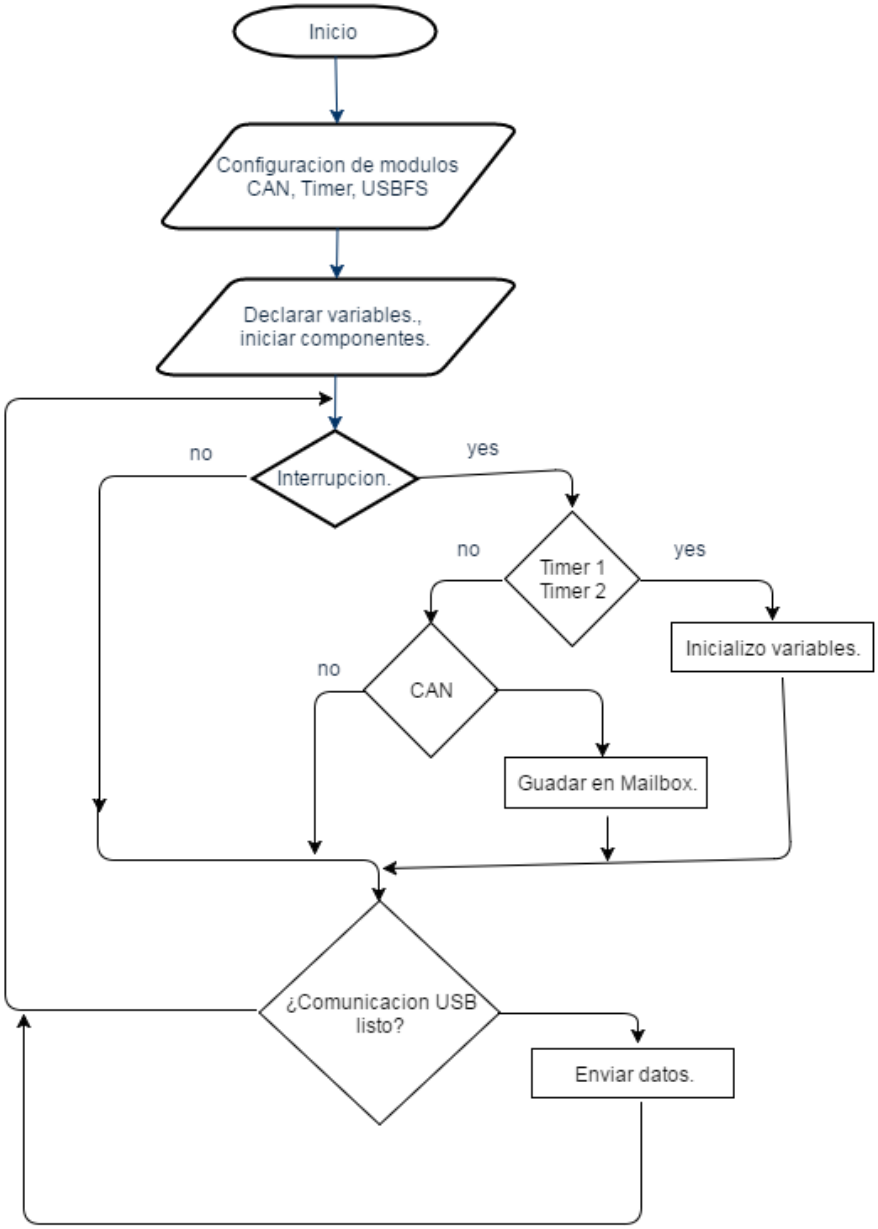

Figura 3.12 Diagrama de flujo del firmware.

El diagrama de flujo representa de manera general la programación del firmware, en donde existen interrupciones por medio de temporizadores en cada determinado tiempo y también por el modulo CAN cada vez que recibe información en el Mailbox. La comunicación USB se establece cada vez que el USB está disponible para enviar. De otra manera, se mantiene en espera hasta que el dispositivo esté listo. Teniendo el firmware, se procede a crear la interfaz en la computadora.

Para la creación de la interfaz, se utilizará el lenguaje de programación de C Sharp debido a las herramientas gratuitas en la red y el conocimiento del lenguaje C. Para establecer una comunicación USB-HID con C- Sharp se agrega un complemento al proyecto llamado CYUSB.dll que permite utilizar funciones para enviar y recibir datos. Posteriormente se diseña la interfaz que a continuación se presenta:

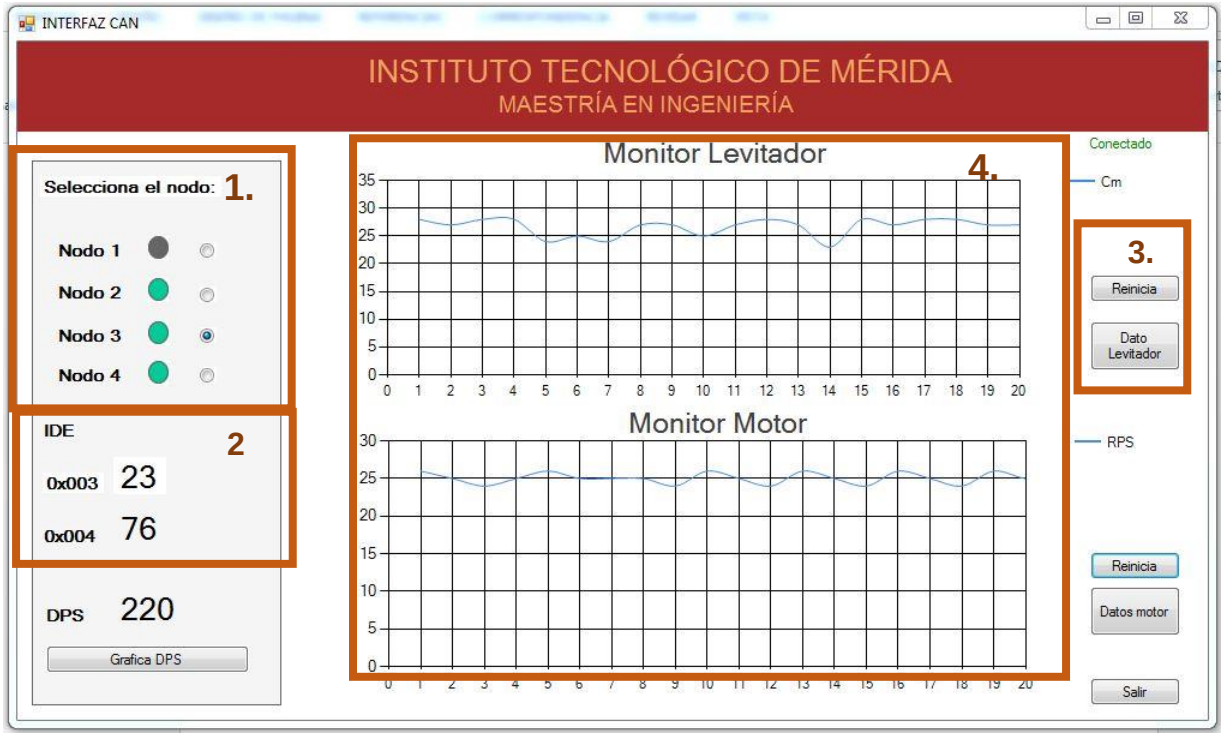

Figura 3.13 Interfaz CAN.

En la interfaz de la Figura 3.13 permitirá observar el comportamiento de los sistemas del motor y levitador, así como el estado de la red CAN. En el recuadro 1 se podrá observar cuando un nodo está activo en la red, cada vez que se reciben datos de la red cada indicador se pondrá en verde. Si se desea observar los datos que se reciben en tiempo real solamente se tiene que seleccionar el nodo a observar. En el recuadro 2 se mostrará la IDE del nodo seleccionado, así como los datos por segundo (DPS) recibidos. En el recuadro 4 se pueden visualizar los sistemas en tiempo real (con un retardo). El levitador que censa la distancia del disco con el sensor determinado. El motor muestra las revoluciones por segundo. En el recuadro

3 muestra 2 botones, una reinicia la gráfica en tiempo real y la otra permite la obtención de los datos graficados en un archivo Excel (Figura 3.15). Lo anterior será de mucha utilidad para la identificación de sistemas con datos experimentales. Dichas opciones del recuadro 3 son disponibles para los dos sistemas graficados.

Figura 3.15 Datos en Excel con la interfaz.

Otras de las opciones disponibles es poder graficar los datos recibidos por cada nodo seleccionado, y también enviar esos datos en un archivo Excel.

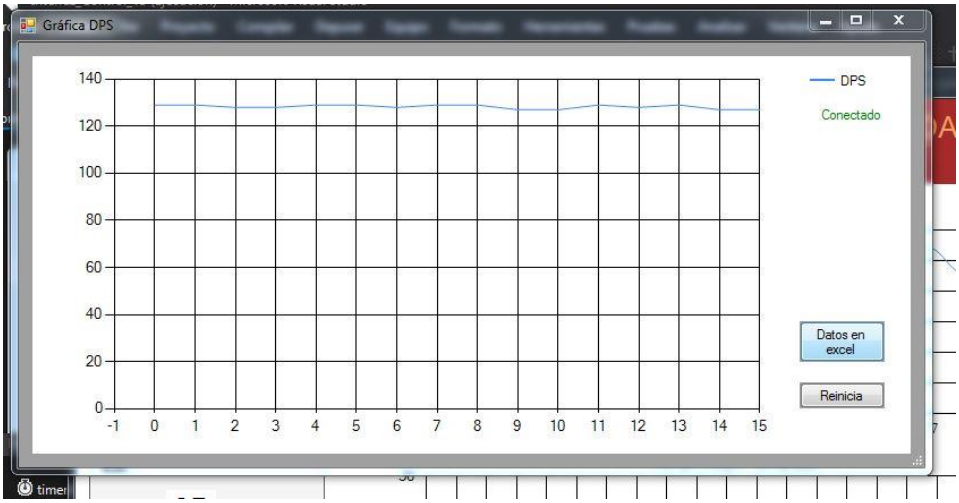

Figura 3.14 Interfaz DPS.

Con la interfaz de la Figura 3.14 servirá para comparar los datos en un tiempo determinado para saber si hay pérdida de datos. Teniendo la interfaz de la Figura 3.13 el monitor está listo para su uso. Para poder comprender la lógica utilizada en la interfaz se expone el siguiente diagrama de flujo.

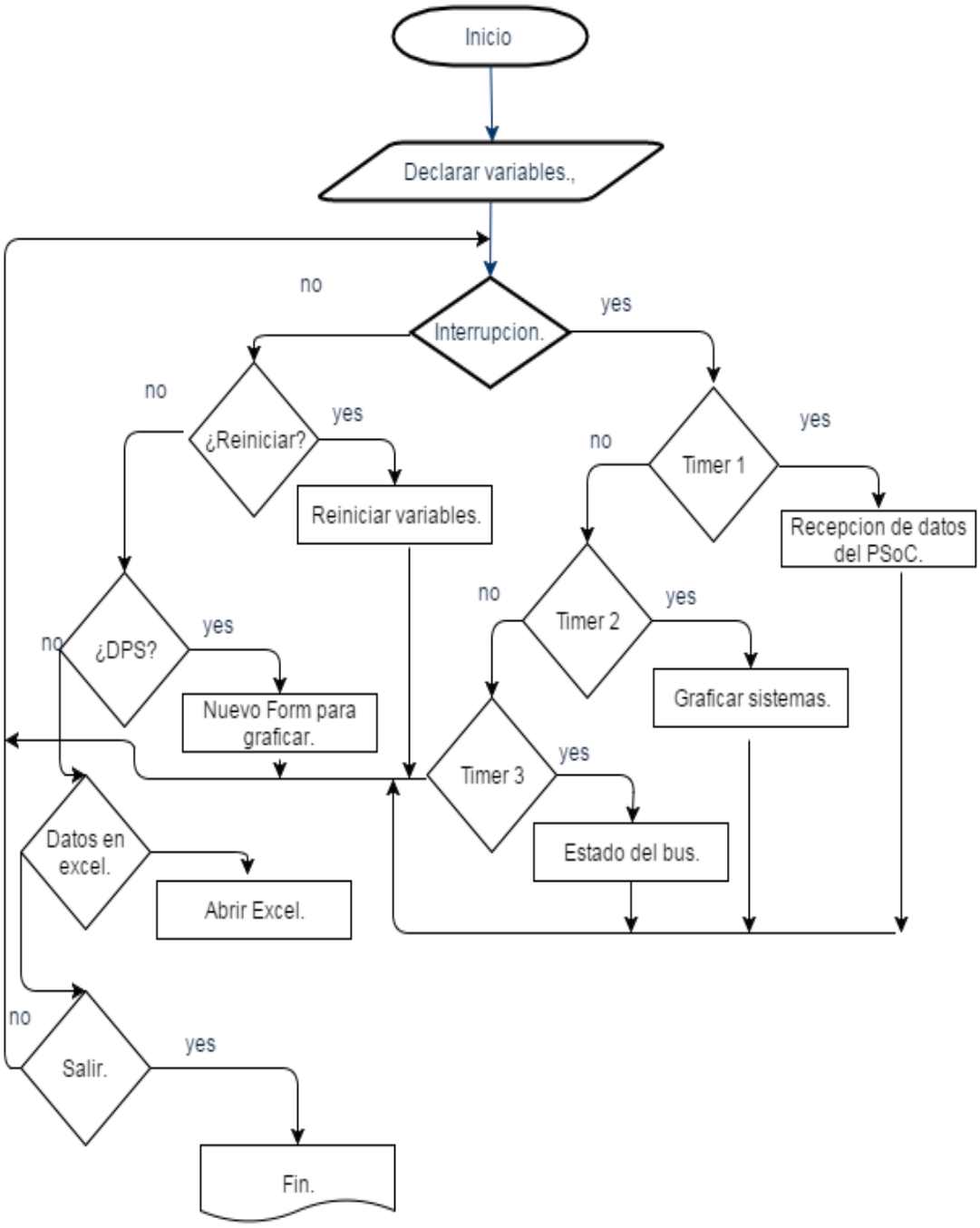

Figura 3.16 Diagrama de flujo de interfaz.

En la Figura 3.16 se presenta el funcionamiento general de la interfaz de la Figura 3.13. La programación en el lenguaje C Sharp es orientada por eventos, por lo que cada objeto en la interfaz tiene programada una acción definida. La interfaz cuenta con temporizadores, cada uno con un tiempo establecido. Los temporizadores servirán para poder graficar los valores obtenidos del bus y de los sistemas. El programa terminará hasta que se pulse el botón cerrar.

3.4 Hardware del motor.

Para la operación del motor se implementó hardware necesario, ya que el motor debe de tener un circuito que haga funcionar el freno magnético y al mismo tiempo debe de tener un circuito que otorgue al motor la corriente necesaria.

El motor va a estar controlado por un CY8KIT-059, aunque solo podrá activar y leer dato del actuador, ya que el verdadero control y procesamiento estará implementado en el nodo 3.

El motor es un NC5475, que puede ser alimentado de 5 a 24 VDC, el circuito responsable de energizar al motor se muestra en la Figura 3.17, dicho circuito tiene entradas de 5, 24 VDC y entrada para variar la velocidad del motor mediante PWM. Tiene salidas para el motor y salida de 5 VDC. Este circuito es muy útil, ya que además de energizar el motor, energizará el sensor óptico para contar las revoluciones por segundo.

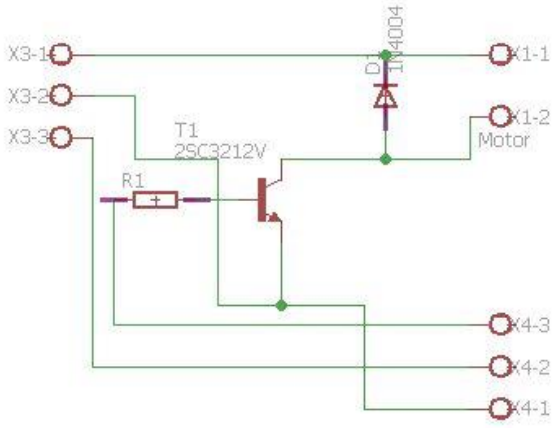

Figura 3.17 Driver para el motor.

Por otra parte el circuito encargado para activar el clutch magnético es mostrado en la Figura 3.18. El circuito activará el clutch magnético con 5.5 VDC

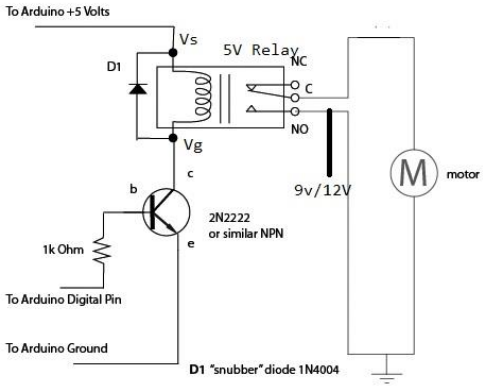

Figura 3.18 Circuito para el Clutch.

Para poder calcular las revoluciones por segundo, se necesita un encoder y un sensor. El enconder obtenido tiene 65 ranuras, por lo que cada 65 cuentas contara como una vuelta del motor. El sensor que servirá para realizar las cuentas es un sensor OptoSwitch, dicho sensor estará energizado por 5 volts. El voltaje va a ser proporcionado por el circuito de la Figura 3.17. A continuación se muestra la implementación de los circuitos expuestos en el motor.

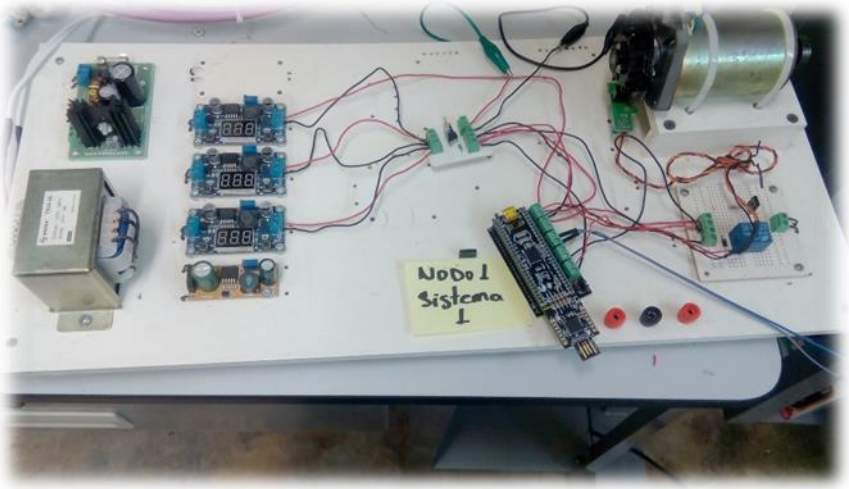

Figura 3.19 Sistema 1 implementado.

3.5 Diseño del controlador en el sistema 1.

A continuación, se diseñará el control del motor ante perturbaciones (frenado) con los siguientes parámetros: El nodo 1 enviará datos cada segundo al nodo de procesamiento que tendrá la tarea de variar el PWM del motor para que ante frenado, el motor mantenga su set-point. El control a diseñar debe responder a no más de 1.5 segundos con un error en estado permanente de 1%, con una tolerancia de sobretiro no mayor al 5%.

Como primer paso, se tiene que obtener la función de transferencia del sistema a diseñar, ya que estos parámetros no son otorgados por el fabricante del motor. Para obtenerlo se identificará el sistema paramétricamente. Como menciona [22] existe un amplio estudio de los sistemas de control, pero sobre todo para la obtención de la función de transferencia a partir del comportamiento de los sistemas a una determinada entrada. Como se explica en el capítulo 2 de este documento, se procede a excitar al sistema con una entrada escalón, y se graficará el comportamiento del sistema para posteriormente clasificar el sistema. En este caso, el escalón será la perturbación del sistema.

En este caso, se aplicará el escalón en 2 situaciones, la primera es fijando una RPS de 25 se le aplica el frenado con el clutch magnético, la segunda es fijar la misma velocidad de 25 RPS cuando está activa el clutch y posteriormente liberar el motor de la carga del frenado. De las dos graficas resultantes se seleccionará la gráfica que se estabilice más rápido [22].

Para la primera caracterización se fija una velocidad del motor a 25 RPS posteriormente se energiza el clutch magnético, que causa la disminución de la velocidad del motor (Figura 3.20) cuyo tiempo de muestreo es de 500 ms.

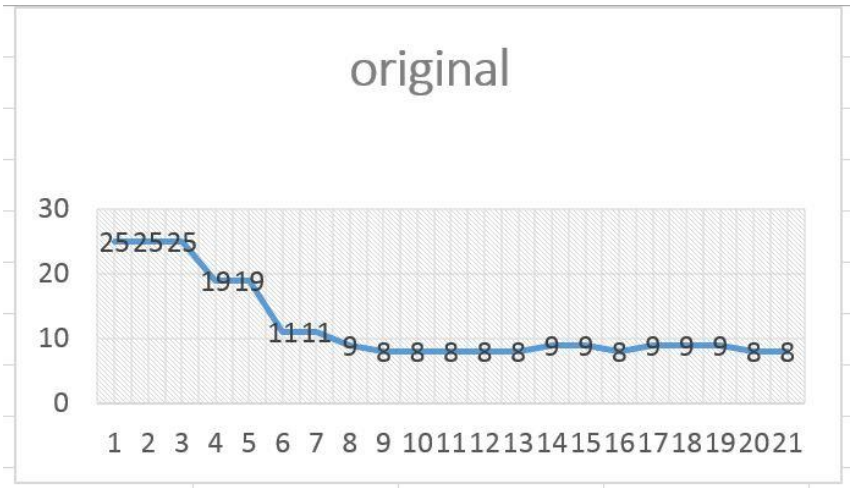

Figura 3.20 Caracterización 1 de motor

Para la segunda caracterización, nuevamente se fija una RPS de 25 cuando el motor está siendo frenado y posteriormente se quita el frenado del clutch. Lo anterior causara el aumento de las RPS en el motor como se observa en la Figura 3.21.

Figura 3.21 Caracterización 2 de motor.

De acuerdo a lo recomendado por [22] se selecciona la caracterización 2 del motor, posteriormente, se debe ajustar la gráfica para poder visualizar el tiempo que toma al sistema estabilizarse. Con lo anterior dicho, se muestra la gráfica ajustada (Figura 3.22), en donde el tiempo de muestreo se ajusta a 1 segundo.

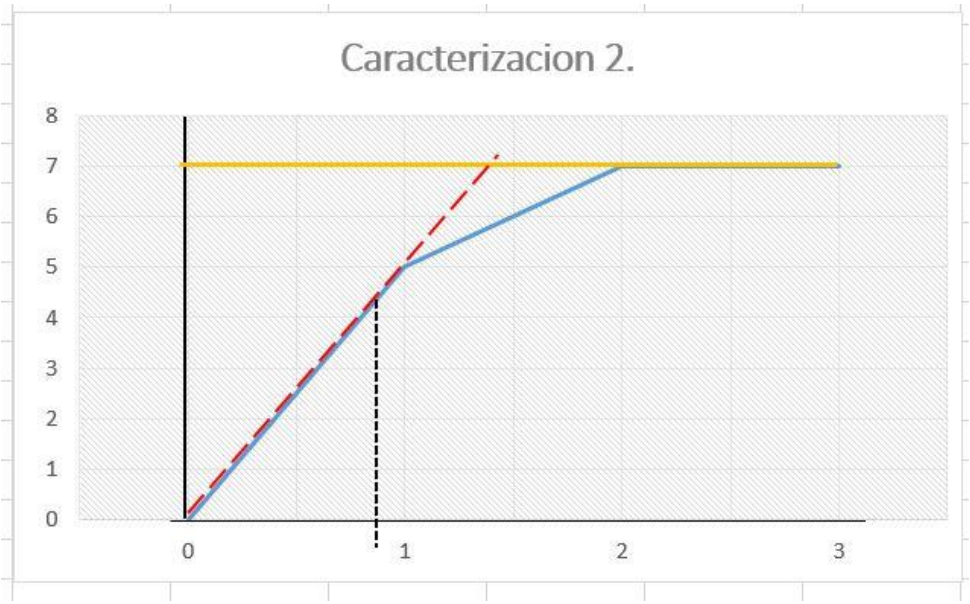

Figura 3.22 Identificación FdT.

De acuerdo a [22] el comportamiento del sistema es sobreamortiguado de primer orden sin retardo, cuya función de transferencia genérica es:

$$G_S = \frac{1}{t_S+1} \quad (1)$$

Como siguiente paso es obtener el valor de t_S colocando una línea recta en la sección de mayor pendiente de la gráfica, dicha línea recta indicará el tiempo de ts cuando la recta intersecte con el valor de $U_{(t)}$. Para los sistemas ideales el tiempo t_S debe corresponder al 63%$U_{(t)}$ [2]. Como el valor de Ts cuando la recta intercede con $U_{(t)}$, es mayor a 63% de $U_{(t)}$, se procede seleccionar el valor de Ts cuando el sistema está en su 63%$U_{(t)}$.

$$T_s = .8S$$

$$G_S = \frac{1}{.8s + 1} \tag{2}$$

Para comprobar si la función de transferencia G_S es similar a la real, con la ayuda de software Matlab y Simulink se procede a comparar valores.

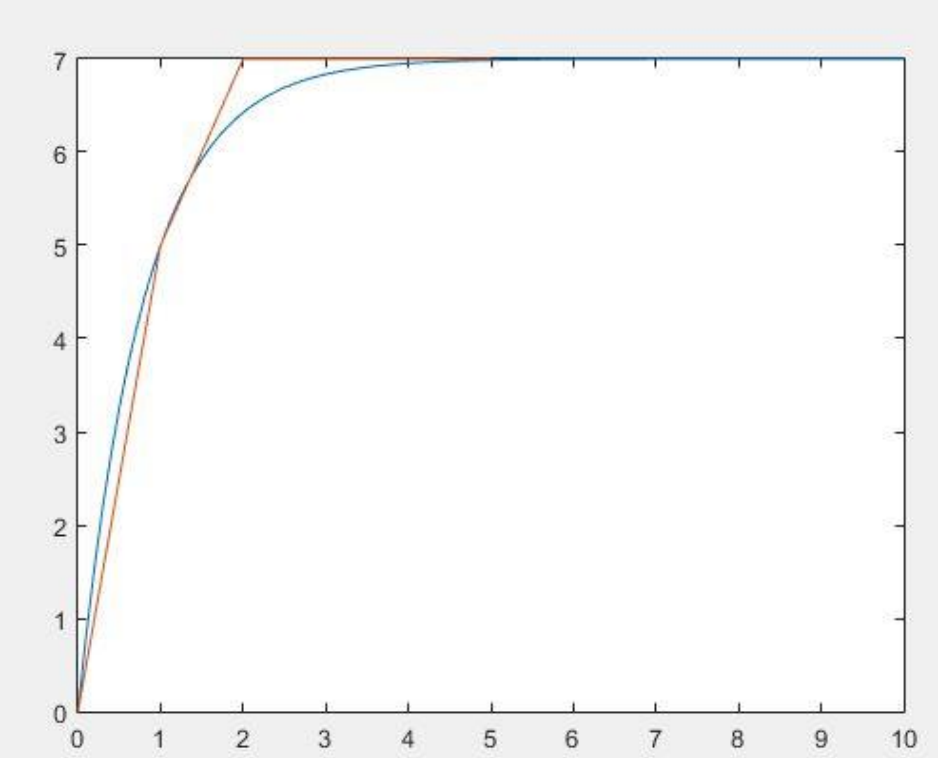

Figura 3.23 Primera aproximación Motor.

Como se observa en la Figura 3.23, la función de transferencia estimada a pesar de su similitud, es más lenta a la real, por lo tanto no es factible, se tendrá que hacer otra aproximación, utilizando los polos reales distintos. Buscando la similitud tanto en estado transitorio como estable de la gráfica en tiempo real, se encuentran los parámetros.

$$G_S = \frac{1}{(.4S + 1)(.4S + 1)} \tag{3}$$

Para comprobar si la función de transferencia estimada sea similar a la forma de la Figura 3.21, con la ayuda del programa Matlab y Simulink, se comparan dichos valores.

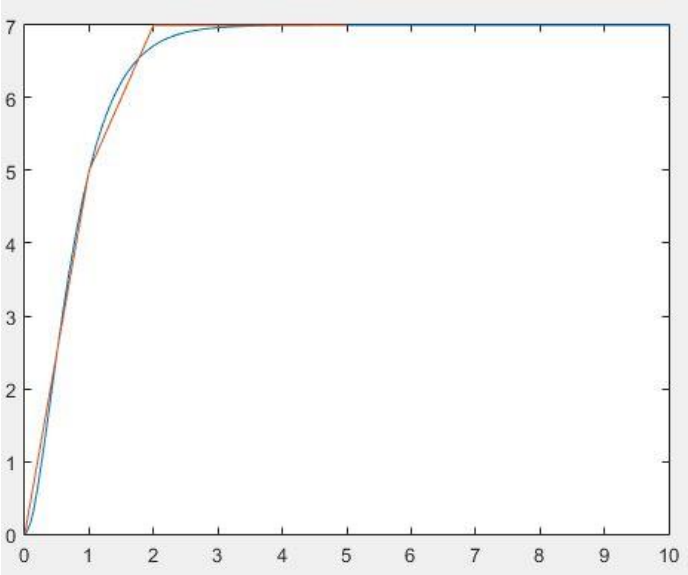

Figura 3.24 Aproximación final motor.

Con los valores de la ecuación 3 se establece la función de transferencia del sistema. Con la función obtenida, el siguiente paso es diseñar el control que satisfaga los requisitos expuestos con anterioridad. Hay que recalcar que el sistema es lento, ya que enviará datos al procesador cada segundo. Para el diseño del control se siguen las recomendaciones de [22], [21], de conocer el error en estado estacionario cuando la función de transferencia está en lazo cerrado.

Calculo de error en estado estacionario en lazo cerrado de FdT.

$$e_p = \lim_{s=0} \frac{S}{1+\frac{1}{(.4S+1).(4S+1)}} * \frac{1}{S} \tag{4}$$

$$e_p = \frac{1}{1+1} = 0.5$$

$$e_p = 50\%$$

Sabiendo el error en estado estacionario en lazo cerrado y siguiendo las recomendaciones de [22], se calcula el valor de un control proporcional (Kp) con un error en estado estacionario de 1%.

$$e_p = \lim_{s=0} \frac{S}{1 + \frac{Kp\ 1}{(.4S+1)(.4S+1)}} * \frac{1}{S}$$

$$e_p = \frac{1}{1 + \frac{Kp}{1}} = .02$$

$$K_p = \frac{\frac{1}{.01} - 1}{1}$$

$$K_p = 99 \tag{5}$$

Para saber si el control proporcional satisface los requerimientos del control, con la ayuda de Matlab, se agrega una Kp=99 a la función de transferencia en lazo cerrado.

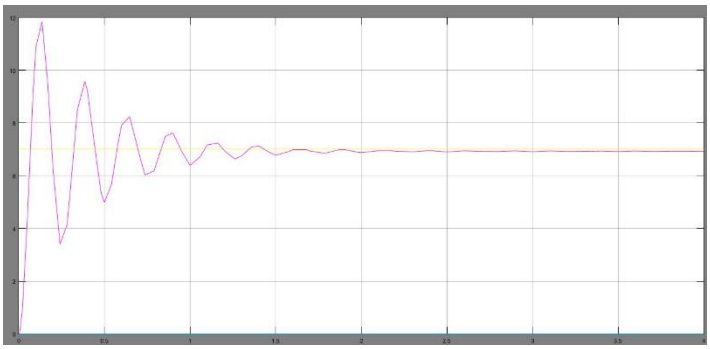

Figura 3.25 Respuesta a control proporcional.

Como se observa en la Figura 3.25 el valor de Kp=99, hace que el sistema tenga sobretiros mayor del 5 %, con lo que no cumple con lo deseado a pesar que en estado estacionario mantenga el 1% de error. Se agregará al sistema un control integral, que mantenga una kp de 99.

Para calcular los valores del control integral respetando los valores de la ganancia calculada, se debe cumplir la siguiente relación de proporción entre cero y polo:

$$\frac{Z_0}{P_0} = 99. \tag{6}$$

Se propone la posición del cero del compensador lo más pegado al origen del plano complejo, y de tantos valores la relación que mayor satisfizo es el valor de Z_0=1.7 $P_0 = 0.017$. Por lo tanto el controlador PI queda:

$$\frac{s+1.7}{s+0.017} \tag{7}$$

Se comprueba los valores para saber si satisface los requerimientos del control deseado.

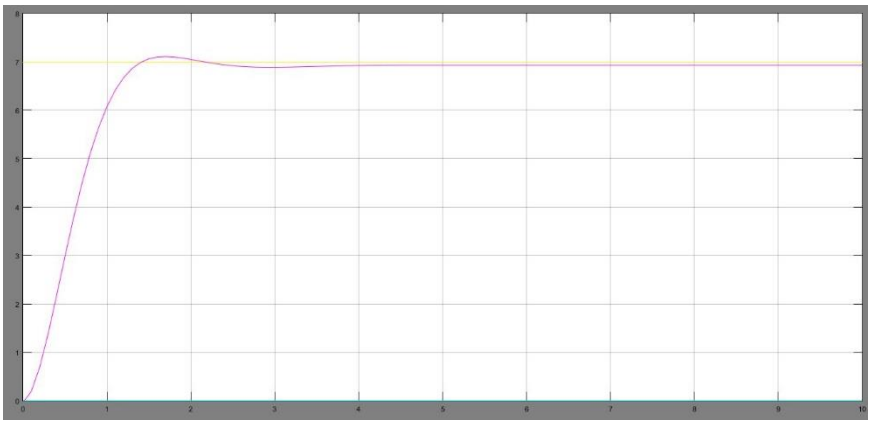

Figura 3.26 Control PI.

Como se visualiza en la Figura 3.26 el control PI satisface los requerimientos del control deseado: responder a no más de 1.5 segundos con un error en estado estacionario de 1%, sin sobretiros que excedan el 5%.

Ahora se convertirán los valores para utilizarlos en un microcontrolador. Como se diseñó el control en el medio continuo, se debe pasar los valores de la función de transferencia y del control al estado discreto, para ello se utiliza el método de discretización de Tustin [21] con el tiempo de muestreo a .1 segundos. Utilizando dicho método las expresiones quedan de la siguiente manera:

$$\frac{1}{(.4S+1)(.4S+1)} \Rightarrow \frac{.0265z+.02243}{z^2-1.558z+.6065}$$

$$\frac{s+1.7}{s+0.017} \Rightarrow \frac{z-0.8301}{z-0.9983} \tag{8}$$

Una vez, traspasado al estado discreto se corrobora si el control no sufre alguna modificación importante.

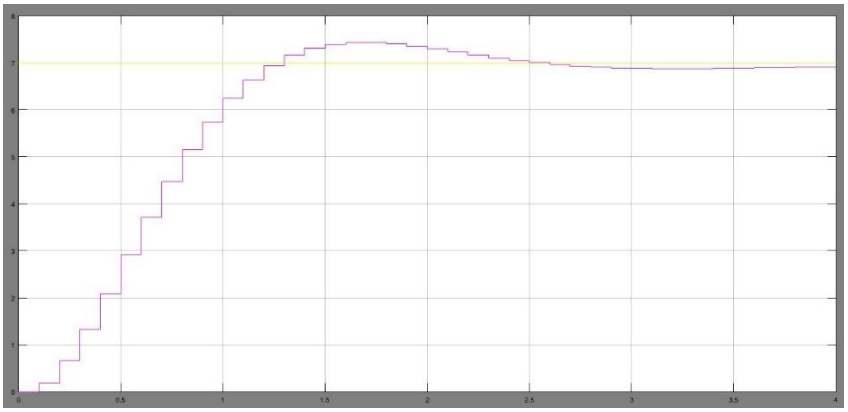

Figura 3.27 Control Discreto.

El siguiente paso es obtener la ecuación de diferencias del control discreto (PI) para poder implementarlo en el microcontrolador:

$$G_s(z) = \frac{U_z}{E_z} = \frac{z - 0.8301}{z - 0.9983}$$

$$(z - 0.9983)U_z = (z - 0.8301)E_z$$

$$U_{(z)} = 0.9983Z^{-1}U_{(z)} + E_{(z)} - 0.8301z^{-1}E_{(z)} \quad (9)$$

Con la ecuación 9 se obtiene la ecuación de diferencias:

$$U_{(k)} = 0.9983\ U(k-1) + E(k) - 0.8301\ E(k-1) \quad (10)$$

3.6 Hardware levitador.

El levitador necesita del perfecto funcionamiento de un ventilador que puede ser energizado hasta 24, Por lo tanto, es necesario de un circuito para poder variar la

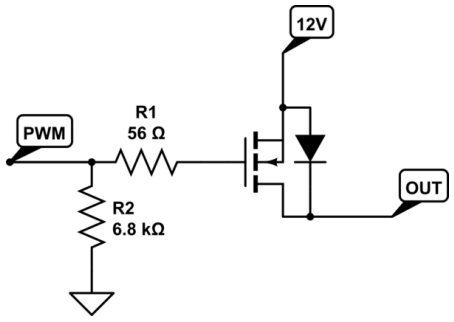

Figura 3.28 Driver Motor sistema 2.

velocidad del motor mediante un PWM.

Con el circuito de la Figura 3.28 se podrá variar la velocidad del motor mediante un PWM del PSoC. El sensor de proximidad que utilizará el sistema es un sensor Sharp GP2Y0A21YK0F, donde su rango de trabajo es de 10-80 cm. La respuesta del sensor es analógica y no lineal, por lo que se tiene que establecer una tabla de valores para cierta distancia en el tubo del levitador.

adc	cm	adc	cm	adc	cm
85	0	43	21	18	42
80	1	41	22	17	43
77	2	40	23	16	44
73	3	39	24	15	45
70	4	37	25	14	46
67	5	36	26	13	47
65	6	35	27	12	48
64	7	33	28	11	49
63	8	32	29	10	50
62	9	31	30		
61	10	30	31		
60	11	29	32		
59	12	28	33		
58	13	27	34		
57	14	25	35		
55	15	24	36		
53	16	23	37		
51	17	22	38		
49	18	21	39		
47	19	20	40		
45	20	19	41		

Figura 3.29 Valor ADC vs distancia.

Los valores de la figura de arriba son los establecidos para las distancias medibles en el tubo del levitador. Como se visualiza el sensor va a medir de 0-50cm de

manera correcta. Para realizar las perturbaciones al sistema se diseñó una tapa capaz de abrir aberturas para que el flujo de aire en la tapa cambie (Figura 3.30).

Figura 3.30 Tapa para perturbaciones.

3.7 Diseño del control sistema 2

Para el diseño del control, Se debe conocer la función de transferencia del sistema, en este caso no se conoce, por lo cual se estimará paramétricamente a partir de datos experimentales.

Identificación Paramétrica del sistema.

De acuerdo a [22] hay que fijar una referencia (setpoint), dicha referencia debe mantenerse durante un determinado tiempo, posteriormente se le agregará una perturbación al sistema, dicha perturbación será la entrada escalón que se le aplicará al sistema para saber su comportamiento. Con el nodo que se encarga en el monitoreo del sistema se grafica su comportamiento ante dicha perturbación.

Figura 3.31 Gráfica del levitador con ruido .

Figura 3.32 Gráfica del levitador sin ruido.

En la Figura 3.31 se observa el comportamiento del sistema ante una perturbación, como se visualiza, la señal que es enviada por el sensor infrarrojo tiene ruido, por lo tanto hay que tratar el señal recibida para realizar un mejor análisis de la respuesta.

En la Figura 3.32 se muestra la señal de la Figura 3.31 sin el ruido, por lo que facilitará su análisis. Ahora lo que interesa es observar de cerca el tiempo que le lleva al sistema estabilizarse ante la perturbación, por las razones que se acaban de señalar, del rango de datos que se tiene en la gráfica, solo se seleccionarán los datos del tiempo que lleva al sistema a estabilizarse.

Figura 3.33 Cálculo de la función de transferencia.

En la Figura 3.33 se estima la función de transferencia, hay que recalcar que para obtener esta grafica se tuvo que seleccionar el rango de valores del tiempo en la cual se estabiliza el sistema ante una perturbación. De acuerdo a [22], la Figura 3.33 representa una respuesta sobreamortiguada de primer orden por lo tanto la función de transferencia es $G_S \frac{K}{t_S+1}$ en donde el valor de la constante de tiempo se obtiene sobre la gráfica, para ello se observa el tiempo correspondiente del 63%$U_{(t)}$ [22]:

$$G_s = \frac{1}{1.7s+1} \qquad (11)$$

Con la ayuda del software Simulink de MATLAB se compara la función de transferencia calculada versus los datos de la Figura 3.33, para corroborar si son semejantes:

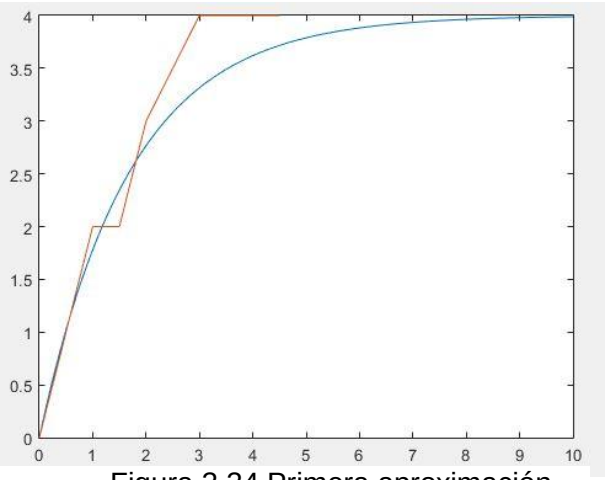

Figura 3.34 Primera aproximación.

En la Figura 3.34, se observa que la función de transferencia calculada (azul) es mucho más lenta que la función de transferencia real (rojo), por tal motivo dicha función no es útil. Se tendrá que realizar otra aproximación, pero ahora de acuerdo a las recomendaciones de [22], se aproximada mediante la técnica de polos reales distintos. Dicha técnica establece que la función de transferencia tendrá la forma:

$$G_S \frac{K}{(T_1 S+1)(T_2 S+1)} \tag{12}$$

En donde T_1 y T_2 son las constantes de tiempo. Para calcular las contantes de tiempo se recomienda utilizar el método de Strejc y posteriormente ajustar las constantes de tiempo hasta obtener los resultados deseados [22]. En este caso como es un sistema de primer orden sin retraso, ajustamos las constantes de tiempo hasta obtener la respuesta más parecida a la real.

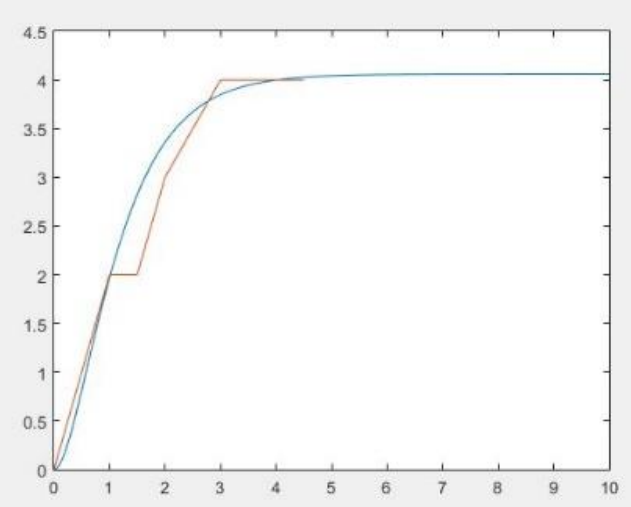

Figura 33.35 Aproximación final

En la Figura 33.35 se observa la aproximación más parecida a la real utilizando la técnica de polos reales distintos, con lo anterior se establece que la función de transferencia del sistema es:

$$G_S \frac{1.015}{(.8S+1)(.45S+1)} \tag{13}$$

Diseño del control.

El control a diseñar debe satisfacer los siguientes parámetros: el control debe tener una respuesta de al menos 2 segundos con un error en estado estacionario del 2%. A continuación se calcula el error en estado estacionario con la función de transferencia en lazo cerrado propuesto por [22].

Error en estado estacionario en lazo cerrado:

$$e_p = \lim_{s=0} \frac{S}{1+\frac{1.015}{(.8S+1)(.45S+1)}} * \frac{1}{S} \tag{14}$$

$$e_p = \frac{1}{1+\frac{1.015}{1}} = 0.49$$

$$e_p = 49\%$$

Se tiene un error en estado estacionario del 49%, nuevamente con la ayuda del Simulink se corrobora dicho calculo (Figura 3.36).

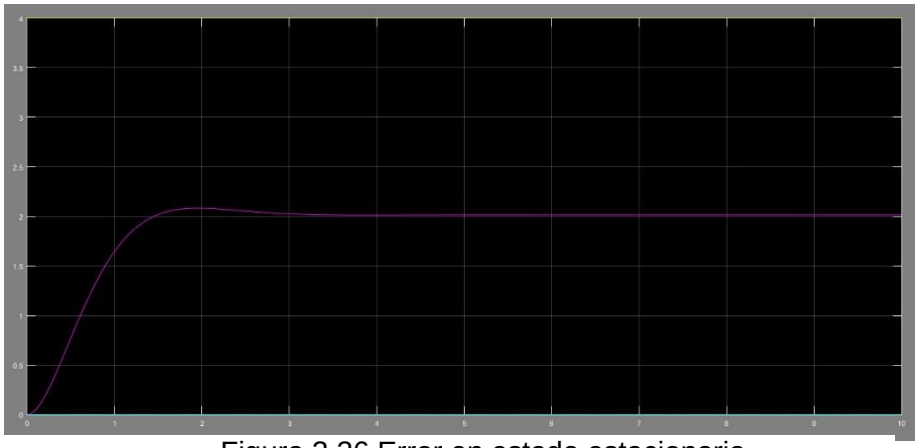

Figura 3.36 Error en estado estacionario.

Estimando la ganancia para quitar el error en estado estacionario:

$$e_p = \lim_{s=0} \frac{S}{1+\frac{Kp\ 1.015}{(.8s+1)(.45s+1)}} * \frac{1}{S} \tag{15}$$

$$e_p = \frac{1}{1+\frac{1.015\ Kp}{1}} = .02$$

$$K_p = \frac{\frac{1}{.02} - 1}{1.015}$$

$$K_p = 48.275$$

Con el resultado de la ecuación 15 se obtiene el valor proporcional (Kp) de 48 para quitar el error en estado estacionario de la función de transferencia en lazo cerrado con un error de 2%. Nuevamente con la ayuda del simulink corroboramos si con un control proporcional puede satisfacer los requerimientos del control deseado.

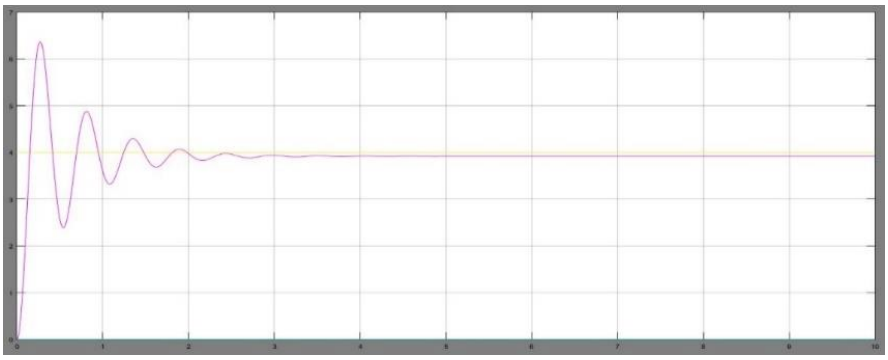

Figura 3.37 Control proporcional.

Como se observa en la Figura 3.37, el control proporcional calculado, genera sobretiros de más del 50%, lo cual no es favorable, por lo tanto, hay que añadir un control integral [22]. Para calcular los valores del control integral respetando los valores de la ganancia calculada, se debe cumplir la siguiente relación de proporción entre cero y polo:

$$\frac{Z_0}{P_0} = 48. \tag{16}$$

Se propone la posición del cero del compensador lo más pegado al origen del plano complejo, y de tantos valores la relación que mayor satisfizo es el valor de Z_0=1.2 $P_0 = 0.025$. Por lo tanto el controlador PI queda:

$$\frac{s+1.2}{s+0.025} \tag{17}$$

Dicha expresión en serie con la función de transferencia en lazo cerrado tiene la siguiente respuesta:

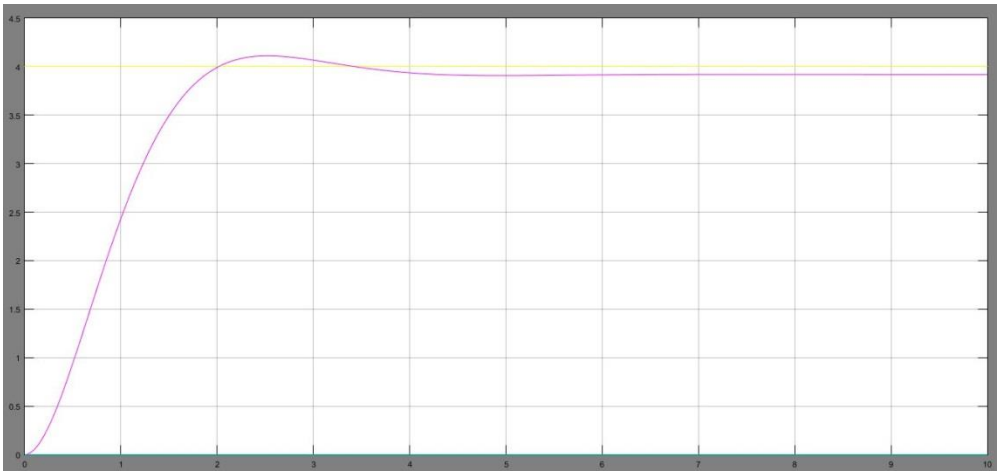

Figura 3.38 Control PI.

Observando la respuesta de la Figura 3.38, el controlador cumple con el funcionamiento deseado del sistema. Ahora se convertirán los valores para utilizarlos en un microcontrolador. Como se diseñó el control en el medio continuo, se debe pasar los valores de la función de transferencia y del control al estado discreto, para ello se utiliza el método de discretización de Tustin [21]. Utilizando dicho método las expresiones quedan de la siguiente manera:

$$\frac{1.015}{(.4S+1)(.45S+1)} \Rightarrow \frac{.01257z+.0112}{z^2-1.683z+.7066} \tag{18}$$

$$\frac{s+1.2}{s+0.025} \Rightarrow \frac{z-0.8801}{z-0.9975} \tag{19}$$

Con lo anterior se corrobora en Simulink si es correcta la conversión

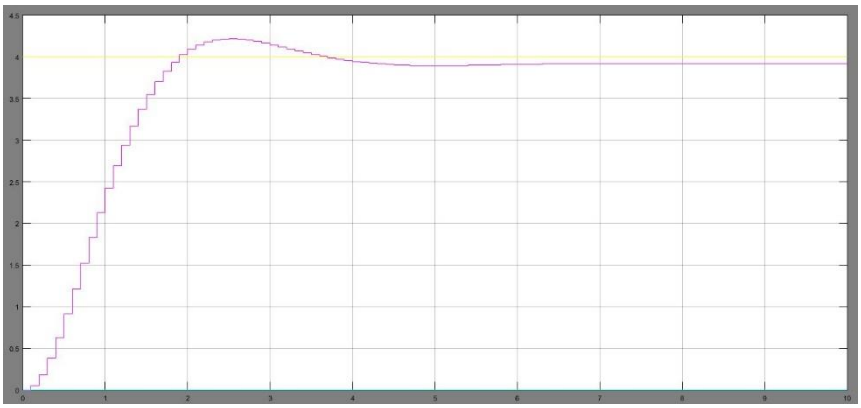

Figura 3.39 Control Discretizado.

El siguiente paso es obtener la ecuación de diferencias del control discreto (PI) para poder implementarlo en el microcontrolador:

$$G_s(z) = \frac{U_z}{E_z} = \frac{z - 0.8801}{z - 0.9975}$$

$$(z - 0.9975)U_z = (z - 0.8801)E_z$$

$$U_{(z)} = 0.9975Z^{-1}U_{(z)} + E_{(z)} - 0.8801z^{-1}E_{(z)} \quad (20)$$

Con la ecuación 9 se obtiene la ecuación de diferencias:

$$U_{(k)} = 0.9975\ U(k-1) + E(k) - 0.8801\ E(k-1) \quad (21)$$

Capítulo 4 Resultados.

A continuación se presentan los resultados obtenidos de cada sistema e interfaz. Como se planteó en el capítulo 3 de este documento el nodo 4 de la red se encargaría de monitorear la red. Al momento de conectar el PSoC 5 a la computadora por medio del puerto USB, el dispositivo instalará el controlador pertinente de forma automática, esta acción lo hará solamente si es la primera vez que se conecta.

Figura 4.2 Nodo 4

Figura 4.1 Enumeración de dispositivo.

Una vez que la computadora reconozca el PSoC como dispositivo HID, se podrá utilizar la interfaz. La figura 4.3 muestra el estado de la interfaz cuando todos los nodos están transmitiendo a la vez.

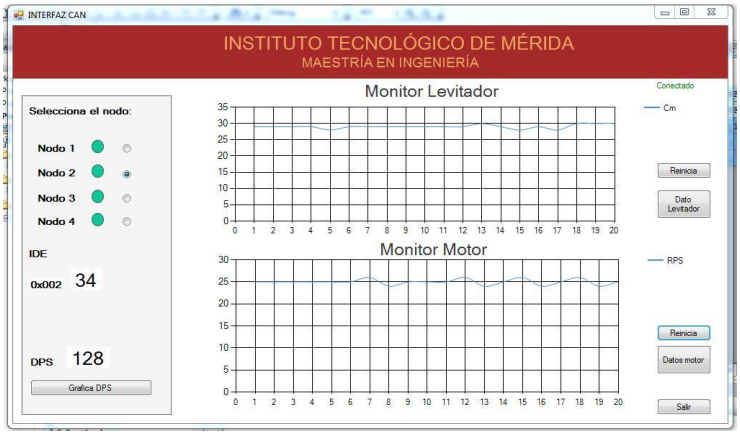

Figura 4.3 Interfaz.

A continuación se presenta el comportamiento de los sistemas cuando están bajo la acción de un control, hay que recordar que el nodo 3 del bus será el responsable del

procesamiento de cada control, los nodos 1 y 2 enviarán datos del sensor al nodo de procesamiento. El procesador enviará la retroalimentación a cada nodo cada determinado tiempo. El bus CAN tiene una velocidad de comunicación de 1 Mbps, ya que cada nodo estará configurado como se presenta en la Figura 3.10 con una implementación Full CAN.

4.1 Resultados del control motor.

En la Figura 4.4 se presenta la implementación real del motor, donde se utiliza el hardware propuesto en el capítulo 3. El identificador para este sistema es el 0x001.

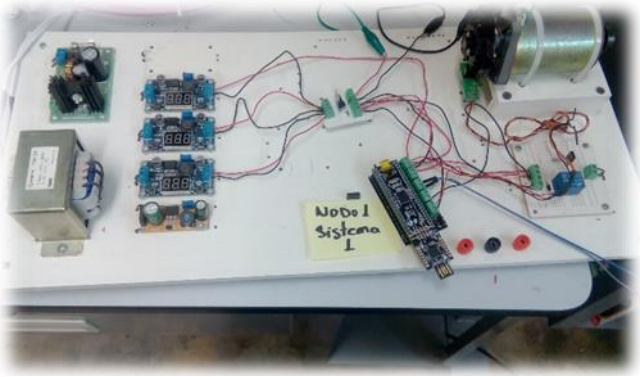

Figura 4.4 Estructura real del motor.

En la Figura 4.5 se observa el comportamiento del motor ante una perturbación (frenado). El procedimiento que se siguió fue el siguiente: previamente se fija un set-point de 25 RPS con el freno no activado. Posteriormente se activa el frenado (segundo 13) por lo que se observa en la gráfica que el motor recupera su velocidad

Figura 4.5 Control en el motor.

en un tiempo aproximado de 2 segundos. Este resultado es muy eficiente, ya que cada segundo el nodo del motor envía datos al nodo de procesamiento.

Dicho comportamiento se puede ver en tiempo casi real (tiene un retardo de 500 ms) o en una gráfica de 50 datos muestreados en un archivo de Excel. Para que la gráfica se pueda apreciar, la Figura 4.5 fue obtenida de un archivo en Excel.

4.2 Resultados control levitador.

A continuación se presenta los resultados del levitador neumático: En la Figura 4.8 se presenta la implementación del levitador en tiempo real, en donde se empotro una estructura cilíndrica hueca transparente (tubo PBC) en una mesa de madera de 70 cm de alto. En la parte inferior se instaló el motor (ventilador) que hará levitar al disco. El identificador para este sistema es el 0x002. En la Figura 4.7 Se muestra el comportamiento del levitador ante perturbaciones, hay que recordar que las perturbaciones son generadas por la tapa del tubo (Figura 3.30). El resultado del control PI es satisfactorio ya que mantiene con un pequeño error el disco en el punto de referencia que es de 36 cm.

Figura 4.8 Levitador en la vida real.

Figura 4.6 Nodo 2.

Figura 4.7 Control levitador.

4.3 Monitoreo de DPS en sistema 1.

La interfaz creada tiene la posibilidad de graficar en cada segundo la cantidad datos transmitidos en un nodo específico. Esto es con el fin de poder saber si existe alguna pérdida de información cuando varios nodos intentan transmitir al mismo tiempo. A continuación se presenta el comportamiento del sistema 1 (motor) cuando solamente el nodo 1 y 3 están transmitiendo datos. Como se comentó anteriormente, el nodo 1 se encarga en recibir los datos de los sensores para posteriormente enviárselo al nodo de procesamiento, luego, después de procesar, los datos son devueltos al nodo 1 para ejecutar un control en el actuador. En la Figura 4.9, presenta una comparación del control aplicado al motor y que posteriormente se le aplica una perturbación. La grafica representa la velocidad en RPS, en donde el eje de las abscisas representa el tiempo en segundos, el de la ordenada los RPS. La señal de

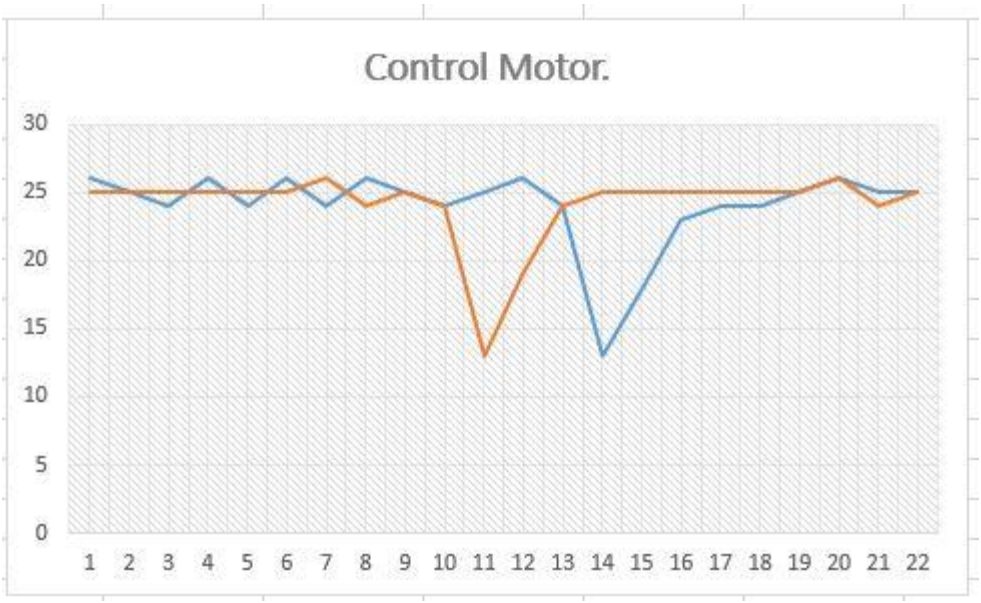

Figura 4.9 Comparación del sistema 1.

color azul representa el comportamiento del sistema cuando todos los nodos están funcionando, la señal de color naranja, representa al sistema cuando solo está en función el control del motor. El momento en que se aplica la perturbación al sistema es diferente en ambos casos, por lo que al comparar las gráficas, el tiempo de disturbio es diferente. Al comparar las señales, se observa que son muy similares, donde varia solamente ±1 RPS.

Dicho lo anterior, ahora se presenta en la Figura 4.10 la gráfica de los datos transmitidos por el nodo 1 cuando solamente el nodo 1 y 3 envía datos en el bus. Posteriormente se graficó el mismo nodo cuando todo los nodos transmiten y resulto una gráfica igual, con lo que se confirma que no hay perdida de información.

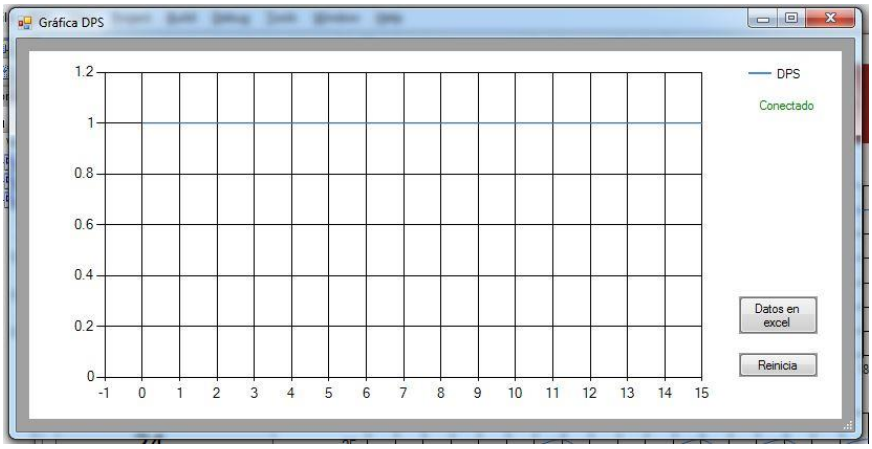

Figura 4.10 DPS sistema 1.

4.4 Monitoreo DPS en sistema 2.

En la Figura 4.11 se presenta la comparación del control aplicado al sistema 2 cuando todos los nodos trasmiten versus cuando solo los nodos 2 y 3 funcionan. La manera en la que se comunican los nodos es la siguiente: el nodo 2, es la encargada de recibir información del sensor infrarrojo, que posteriormente, el nodo 2 envía los datos al nodo 3 (que aplica el control diseñado), después que el nodo 3 procese la información, el nodo le devuelve información al nodo 2, dicha información servirá para controlar el actuador (motor) que hará que levite un disco. En la comparación de las señales de la figura antes dicha, se observa que en la señal donde todos los nodos están trabajando (señal azul) presenta mayor inestabilidad.

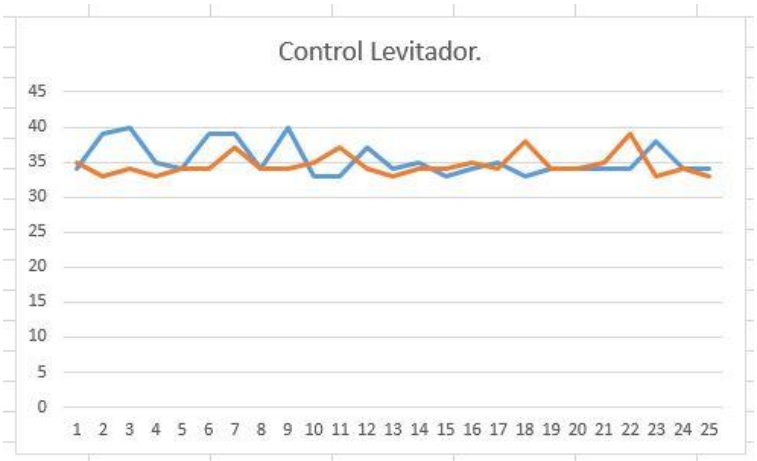

Figura 4.11 Comparación del Control Levitador.

A continuación se presenta una comparación de los datos transmitidos exitosamente por el nodo 2 en los mismos casos que la gráfica anterior. En la Figura 4.12 se presentan dos señales; la azul, representan los datos enviados cuando solamente el sistema 2 está trabajando en el bus. La señal de color naranja, representan los datos enviados cuando todos los nodos están funcionando (sistema 1).

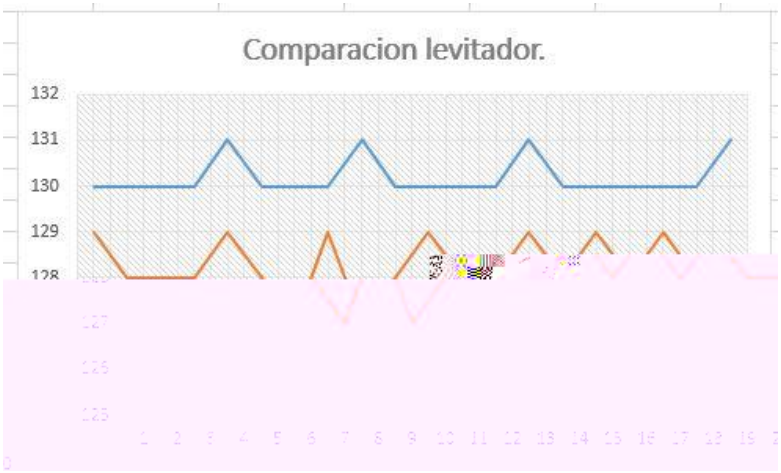

Figura 4.12 DPS Levitador.

En la figura antes mencionada, se observa que la señal naranja obtuvo menos datos enviados al nodo de procesamiento, también se observa que presenta mayor

inestabilidad al enviar datos. La cantidad de datos perdidos alcanzan hasta 4 datos por segundo.

4.5 Monitoreo del nodo de procesamiento.

En la Figura 4.13 se muestra los datos transmitidos exitosamente del nodo 3 de la red CAN. Dicho nodo se ocupa de procesar el control digital para cada sistema, por lo tanto, recibe y envía información a los nodos 1, 2 y 4.

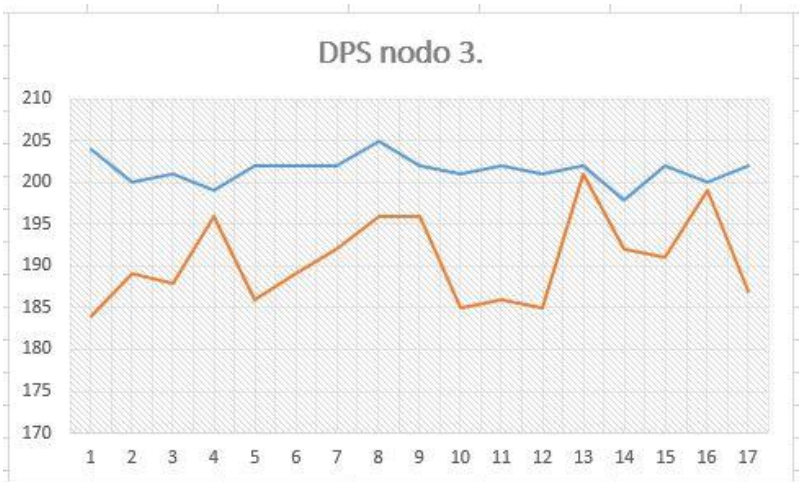

Figura 4.13 DPS nodo 3.

La figura muestra dos señales, una de color azul y otra de color naranja. La primera corresponde a los datos enviados del nodo 3, cuando solamente el sistema del levitador está funcionando. La segunda, corresponde a los datos enviados del nodo 3, cuando los sistemas (levitador y motor) están operando. Evidentemente se observa una disminución de datos enviados a los nodos que activarán los actuadores correspondientes de cada sistema. La diferencia de datos enviados de ambas señales es de ±16 datos. El identificador que tiene asignado el nodo 3 es 0x004, por lo que ante las prioridades de transmisión en el bus, este es el último.

4.6 Acceso al bus en dos nodos.

Para poder realizar un análisis más completo en cuanto a la disponibilidad del bus, se repite la prueba del nodo 2, pero ahora, se le asignará otro identificador. En la tabla se muestra la nueva asignación de los nodos.

Tabla 3 Nueva asignación de identificadores.

Nodo	ID Anterior.	ID nueva.
Nodo 1	0x001	0x002
Nodo 2	0x002	0x001
Nodo 3	0x004	0x004

La Figura 4.14 muestra la comparación de los datos enviados por el nodo 2. Dicha grafica muestra 2 señales. La primera (señal azul) corresponde cuando está en operación el sistema del levitador; la segunda (señal naranja), muestra los datos enviados por el nodo 2 cuando de todos los sistemas propuestos están en funcionamiento. Comparando la gráfica de la figura Figura 4.12 con la actual, se comprueba que existe menos perdida de mensajes transmitidos cuando todos los nodos se encuentran en operación.

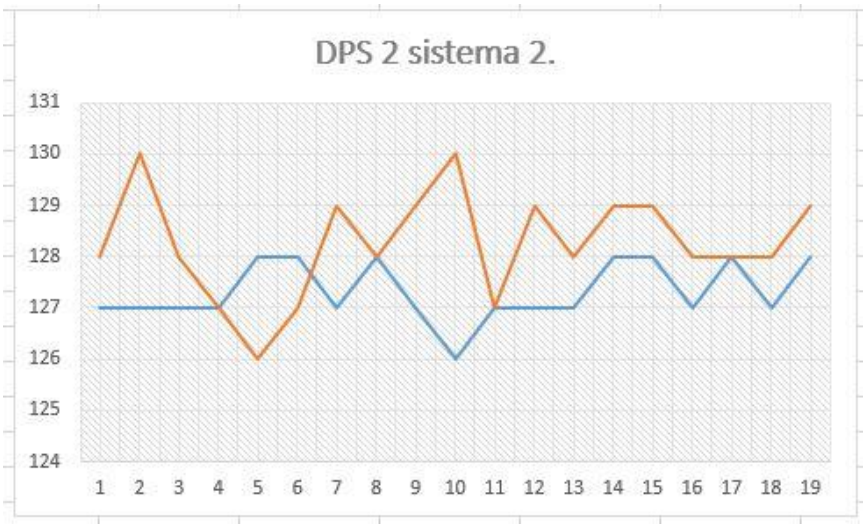

Figura 4.14 DPS 2 Nodo 2.

Capítulo 5 Conclusiones.

El bus de comunicaciones CAN, fue creado para la interconexión de varios dispositivos sin la necesidad de un denso cableado, también para tener una comunicación robusta y económica. Aunque originalmente dicho protocolo fue desarrollado para aplicaciones automotrices, en la actualidad ha incursionado en varias áreas de la ingeniería. El bus de comunicaciones puede estar clasificado de acuerdo a la implementación del Mailbox o por el tipo de trama e inclusive por el tipo de hardware embebido en el chip. Este trabajo hace el enfoque en la implementación del Mailbox, en especial la implementación Full-CAN, como se presentó en los capítulos anteriores, se desea mostrar el comportamiento de la red Full-CAN en diferentes sistemas de control, así como saber el impacto en los sistemas la congestión de datos en el bus.

El microcontrolador utilizado para implementar los sensores, actuadores y transceptores fue el CY8C5888LTI-LP097 en el kit CY8CKIT-059 de la empresa CYPRESS cuya IDE (en inglés Integrated Development Environment) es el PSoC Creator. De manera general el uso del microcontrolador en este trabajo se concluye:

- El Psoc creator facilita la configuración de los módulos CAN, ya que cuenta con un wizard, que reduce la cantidad de código a escribir (Figura 3.9). dicho lo anterior, la implementación Full-CAN y la asignación de los identificadores para cada nodo fue realizada con éxito. Una de las ventajas de realizar la implementación Full-CAN en el PSoC, es el ahorro de tiempo y código, ya que al seleccionar el Mailbox como Full, automáticamente se establecen valores que harán funcionar la implementación (El identificador se establece por el usuario). Por otro lado si es Basic-CAN, se tendrá que configurar las interrupciones e ID´s vía código en el PSoC Creator.
- En la configuración de la velocidad de transferencia en la comunicación CAN a 1Mbps, es importante tener un reloj externo con un error menor o igual a .2%, si no es así, se corre el riesgo de que varios mensajes no puedan ser transmitidos.

La identificación paramétrica en los sistemas a partir de datos experimentales toma un papel muy importante en las aplicaciones en donde se requiere un control, ya que en muchas ocasiones no se cuenta con la función de transferencia real para diseñar un compensador. En el diseño de los controladores de los sistemas 1 y 2 sugeridos en este documento se llegaron a las siguientes conclusiones:

- Es importante escoger los sensores adecuados para la adquisición de datos del sistema a identificar, ya que una mala lectura, podría generar una función de transferencia no conveniente.
- Al momento de excitar al sistema con una perturbación, se tiene que observar si en algún momento dicho sistema se estabiliza, si no es así, utilizar este método puede no ser confiable.
- Al momento de estimar le función de transferencia del sistema, se tiene que comprobar que dicha estimación, sea lo más parecido a la gráfica real, ya que si no es así, al momento de implementar el control puede que no cumpla con los requisitos deseados. En la identificación del sistema realizado, se observó que las funciones del trasferencia no fueron exactamente iguales, por lo tanto los controles diseñados, aunque en la teoría son muy confiables, en la vida real presentan pequeños errores que no afectan los objetivos del control planteado. Si en el momento de implementar el control no cumple con los objetivos planteados, se tiene que repetir todo el procedimiento de identificación o en su caso ajustar los valores. Dicho ajuste es mínimo, ya que los valores calculados son próximos al valor ideal.

En el capítulo 4 se presentaron los resultados de los controles diseñados y la disponibilidad de los nodos al comunicarse entre sí y se observó que en ciertos nodos, la cantidad de datos enviados cambia (Figura 4.12). En el sistema del levitador, se observó que, cuando todos los nodos están en uso, la cantidad de mensajes enviados disminuyen, dicha disminución, no afecta en gran manera al control (Figura 4.11). En contraste, en la Figura 4.14 se presenta el mismo sistema con los identificadores modificados (Tabla 4.1). La explicación a la diferencia de datos enviados del nodo 2, es que al momento en que dos nodos pelean el acceso al

bus para enviar datos, el nodo con el identificador con el número más pequeño (mayor prioridad) gana el acceso al bus [23] [1], lo anterior mencionado, se confirma con las señales presentadas en la Figura 4.13 que corresponden al nodo 3 cuyo identificador es 0x004.La figura muestra que el nodo 4 es el que pierde más datos cuando todos los nodos están en funcionamiento. Por lo tanto, si se desea implementar varios sistemas de control en una red con implementación Full-CAN se recomienda asignar los identificadores más pequeños a los nodos que necesiten enviar información de manera más rápida.

Referencias

[1] W. Lawrenz, CAN system engineering from theory to practical applications, New York: Springer, 1997.

[2] L. C. &. Y. Guo, «Design of the Distributed Control System Based on CAN Bus,» de *Computer and Information Science*, Xi'an, Canadian Center of Science and Education, 2011, pp. 83-89.

[3] J. F. G. C. e. a. M. Garcia Juarez, «Estudio de estrategias de control PI disparado por eventos para sistemas basados en red,» *Congreso Nacional de Control Automatico,* pp. 447-453, 2015.

[4] F. Lars-Berno, «CAN FOR CRITICAL EMBEDDED AUTOMOTIVE NETWORKS,» *IEEE MICRO,* p. 8, 2002.

[5] Y. R. A. F. T. F. A. R. A.-A. H. F. Othman, «Controller Area Networks: Evolution and Applications,» *IEEE,* pp. 3088-3094, 2006.

[6] P. Z. L. Hou, «Implementation of CAN bus device driver design Base on Embedded System,» *IEEE,* p. 4, 2010.

[7] S. N. S. D. E.Giri Prasad, «Real-Time System Bus Architecture for Small Aircraft Using Can-Protocol,» *International Refereed Journal of Engineering and Science (IRJES),* pp. 24-29, 2014.

[8] M. P. S. W. ChunHua Zeng, «CAN Bus Communication System Based On SOC Technology,» *IEEE,* pp. 322-325, 2010.

[9] B. X. YunxiaJiang, «Design and Implementation of CAN-Bus Experimental System,» *IEEE,* pp. 655-659, 2011.

[10] C. L. C. J. PIAO Chang-haol, «A Design for Controller Area Network Bus Real-time Monitoring System,» *IEEE,* pp. 1621-1624, 2011.

[11] J. A. S. Rios, Desarrollo de una red para instrumentacion electronica basada en el protocolo CAN uitlizando microcontroladores, Ciudad de Mexico: Instituto Politecnico Nacional, 2012.

[12] C. L. a. F. Luo, «A Co-Simulation-and-Test Method for CAN Bus System,» *Journal of Communications Vol. 8,* pp. 681-689, 2013.

[13] F. ORTIX, «USE OF CAN BUS IN THE VEGA LAUNCHER AUTONOMOUS

TELEMETRY SYSTEMS,» TEMIS, 2013.

[14] M. Nirubama, «Data Acquisition System Based on CAN Bus and ARM,» Department of ETC, Bharath University, Chennai-73, India, Chennai-73, India, 2014.

[15] H. Y. C. S. Qingbo Yuan, «CAN bus application in deep-sea drilling measurement and control system,» IEEE, Hangzhou Dianzi University Hang Zhou,China, 2014.

[16] J. A. O. A. J. S. G. Carlos Lujan Ramírez, «CONTROL DE MOTOR DC MEDIANTE EL PROTOCOLO DE COMUNICACIÓN CAN CON INTEGRADOS DE LA FAMILIA DSPIC30F DE MICROCHIP,» CONIEEM, Merida, Yucatan, 2014.

[17] J. A. O. Arana, «COMUNICACIÓN DE UNA RED CAN A LA PC UTILIZANDO LA COMUNICACIÓN USB-HID.,» Instituto Tecnológico de Mérida, Merida Yucatan, 2015.

[18] J. Lepkowski, «EMI/ESD Protection,» On semiconductor, Arizona, 2004.

[19] j. a. o. arana, CARACTERIZACIÓN DE UNA RED CAN CON INTERFAZ HID UTILIZANDO MICROCONTROLADORES DE LAS EMPRESAS CYPRESS Y MICROCHIP, Merida: Tecnologico de Merida, 2014.

[20] MICROCHIP, «CAN MODULE,» de *DSPIC*, MICROCHIP, 2007, p. 74.

[21] M. E. L. Guillén, «Identificación de Sistemas: Aplicación al modelado de un motor de continua.,» Departamento de electronica, 2011.

[22] D. A. F. Novelo, «Obtención de la Función de Transferencia de Sistemas mediante la Identificación Paramétrica a partir de datos Experimentales,» de *Apuntes de control*, Merida Yucatan, 2014.

[23] Y. Q. Z. Yu, Control y Monitorización de Plantas Mediante Una Red CAN, Barcelona: Universidad Politecnica de Cataluña, 2011.

[24] Cypress, «Cypress,» 20 08 2016. [En línea]. Available: http://www.cypress.com/knowledge-base-article/difference-between-full-can-and-basic-can-mailbox-kba86565.

[25] A. Bailey, «CAN BUS,» Washington State University, Washington, 2015.