Evaluación de la Desincronización de 3 Placas Choriboard IV

1. Objetivo

Como parte de las necesidades de implementación de un sistema redundante de tiempo real, se requiere que todos los nodos estén de acuerdo en una base de tiempos común. El objetivo de este experimento es observar las diferencias existentes entre los respectivos clocks de las computadoras de vuelo. Para ello, se hizo el experimento que aquí se presenta, el cual consiste en tomar mediciones de los timers de 3 computadoras de vuelo Choriboard IV conectadas a un mismo bus de comunicaciones CAN. Cada una de las computadoras cuenta con un clock local implementado con uno de los timers de su correspondiente microcontrolador. A partir de las mediciones recolectadas, se hicieron gráficos y se obtuvieron conclusiones acerca la necesidad del uso de un algoritmo de resincronización periódica para el sistema final.

2. Introducción

En la computadora de vuelo Choriboard IV, la tolerancia a fallas se implementa a través de la comparación de resultados. Cuando ocurre una diferencia de resultados, se asume que ocurrió una falla. Debido a que el sistema es de tiempo real, no solo es importante el valor del resultado, si no que además el momento en el que se obtiene dicho resultado. Esto último es lo que obliga a que los nodos estén de acuerdo en una base de tiempos.

Otro de los motivos por los que es importante, para evitar las colisiones en el bus de comunicación. Si todos los nodos coinciden en la base de tiempos, se pueden evitar las colisiones y todos los nodos pueden acceder al bus de forma ordenada para cumplir con los tiempos requeridos en el intercambio de información. Esto es una característica que simplifica los sistemas redundantes, la comunicación a través de un bus, por sobre muchas comunicaciones 1 a 1 [1].

3. Descripción del Experimento

El experimento consiste en tomar mediciones de los clocks internos de cada una de las placas y observar las diferencias que se manifiestan. Cada una de las computadoras de vuelo implementa un clock local a través del uso de alguno de los timers de su correspondiente microcontrolador. Una cuarta placa denominada *master* le solicitará periódicamente a todas las computadoras de vuelo que realicen una captura de sus respectivos clocks locales. Esto se denomina timestamp. En la figura 1 se muestra un esquema.

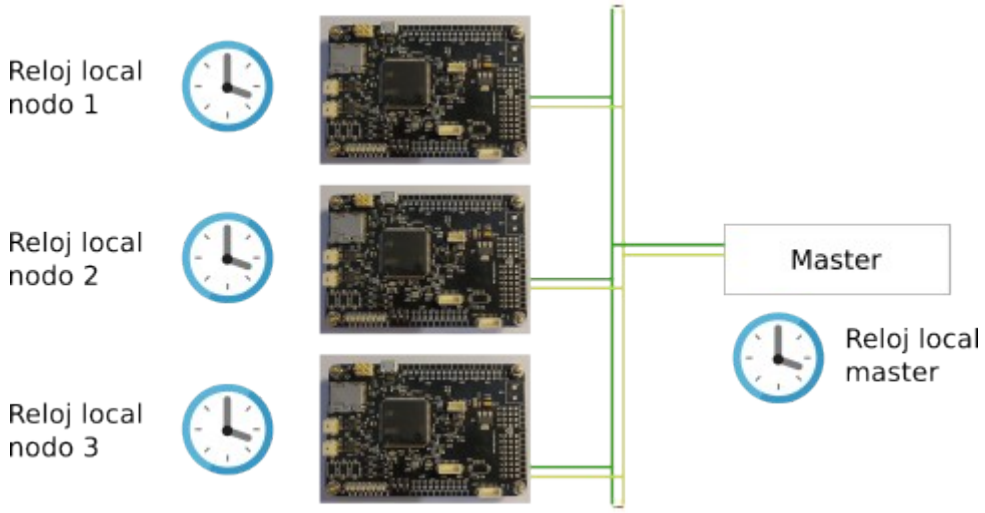

Figura 1: Configuración de conexiones entre las computadoras de vuelo y la placa master. La conexión se realiza con dos cables que conforman la comunicación CAN. En los extremos del bus se colocan resistores de terminación de 120 Ohms. Cada nodo tiene su respectivo reloj local a sí mismo.

Para dar comienzo al experimento, el master debe enviar un mensaje a través del bus CAN, avisándole a los 3 nodos que estos deben iniciar sus respectivos clocks locales. Debido a que las tres computadoras de vuelo se encuentran conectadas al mismo bus, estas detectarán el mensaje recibido al mismo tiempo. En consecuencia, ejecutarán la acción de iniciar sus respectivos clocks internos.

Una vez que las 3 computadoras de vuelo se encuentren con sus respectivos clocks corriendo, el master enviará periódicamente un mensaje al bus CAN. En el momento en el que las computadoras de vuelo detecten que recibieron este mensaje, estas toman un timestamp, empaquetan ese dato en un mensaje de CAN e inyectan dicho mensaje en el bus para que el master lo reciba. El master a su vez hace de pasamanos y envía todas las respuestas recibidas a través de su interfaz USB, la cuál estará conectada a una PC a través de un cable micro USB. Finalmente, todos los datos enviados por USB hacia la PC son recolectados y guardados en un archivo, utilizando un script de Python.

Para finalizar el experimento, el master debe enviar otro mensaje distinto, el cuál será interpretado por todos los slaves para decirles que detengan sus clocks internos.

Un aspecto que puede llamar la atención a priori, es el hecho de que el master periódicamente envía un mensaje para solicitar el timestamp de cada computadora de vuelo. Un error fácil de cometer es pensar que esto podría no ser necesario, y que cada computadora de vuelo puede enviar el mensaje con su propio timestamp periódicamente por su cuenta. El problema aquí es que ese período será calculado por cada computadora de vuelo, pero lo harán tomando como base su propio reloj local, es decir, el nodo 1 estaría enviando un mensaje con su timestamp de forma periódica, tomando como base de tiempos su propio reloj. Lo mismo para los nodos 2 y 3. Por un lado, esto resulta incorrecto ya que los 3 nodos no están de acuerdo en una base de tiempos común. Pero más allá de esto, los 3 nodos entregarán mediciones que serán exactamente iguales. Esto no será representativo de la realidad, debido a que cada nodo hizo la medición tomando como base de tiempos su propio reloj. Para observar resultados, es necesario que las placas utilicen la misma base de tiempos.

Otra posibilidad, podría ser realizar el experimento al revés, es decir, que los nodos envíen un mensaje de forma periódica al master y que cuando este los recibe, les aplique un timestamp. El problema es que debido a que es un bus, solo puede existir un solo mensaje a la vez en este. Esto quiere decir que si los tres nodos se encuentran perfectamente sincronizados, estos querrán enviar un mensaje por el bus CAN, pero como solo puede existir uno a la vez en el bus, el master no puede recibirlos al mismo tiempo y les aplicará timestamps distintos, cuando en realidad los nodos sí enviaron el mensaje al mismo tiempo. Este análisis junto con el del párrafo anterior, dejan como única posibilidad para realizar el experimento, el hecho de que el master sea quien consulte periódicamente por el tiemstamp de las computadoras de vuelo.

4. Desarrollo

Como nodo master se utilizó una placa de desarrollo del fabricante ST, la placa NUCLEO-F746ZG, junto con un módulo de CAN transciever, el SN65HVD230. Este mismo transciever es el que se encuentra en las placas Choriboard IV. Los módulos SN65HVD230 cuentan con el resistor de 120 Ω integrado, por lo que en el experimento solo se agregó un resistor de terminación extra. En la figura 2, se muestra una imagen con la placa master y los 3 slaves, conectados al mismo bus CAN.

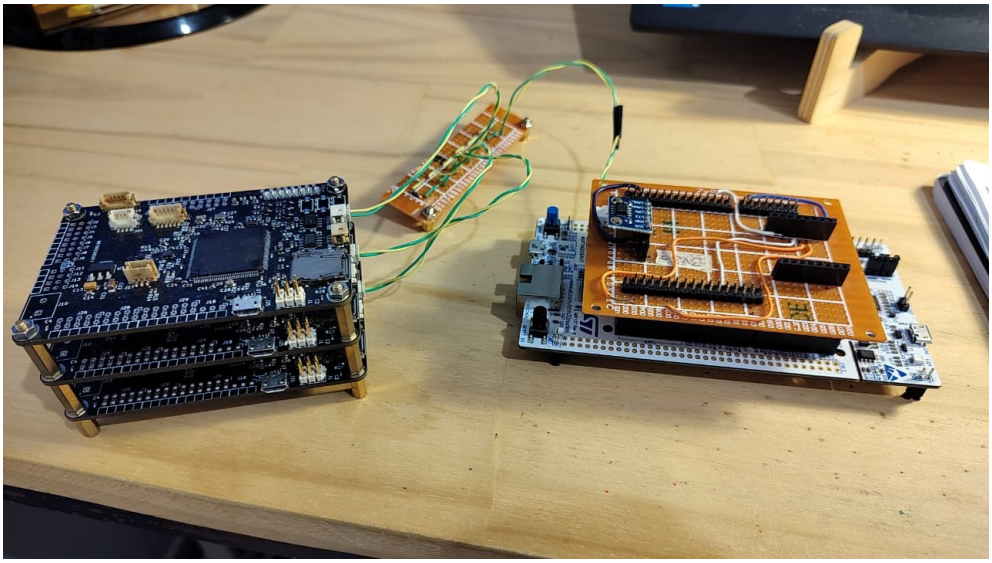

Figura 2: A la izquierda las placas Choriboard IV y a la derecha la placa master y su módulo CAN transciever. Todas se encuentran conectadas al mismo bus CAN. El cable verde corresponde CAN-L y el amarillo a la línea CAN-H.

Para poder llevar a cabo el experimento, se desarrolló un firmware sencillo para el nodo master y otro para los 3 nodos Choriboard IV (las computadoras de vuelo cuentan exactamente con el mismo firmware).

Como reloj de las placas slave, se utilizó uno de los timers integrado en el microcontrolador. Durante la etapa de desarrollo de las computadoras de vuelo, cada timer fue reservado para un uso específico, por ejemplo control de salidas PWM. Se tuvo que seleccionar alguno de que no estuviera reservado, el timer 3. Este timer de 32 bits fue configurado para tener una frecuencia de 1 MHz, es decir que su contador se incrementa en 1 cada 1 microsegundo. Como fue mencionado, las 3 placas slave cuentan con el mismo firmware, por lo que esta configuración de timer se aplicó para las 3 placas slave por igual.

4.1. Esquema de Comunicación

Se muestra un diagrama secuencial, figura 3.

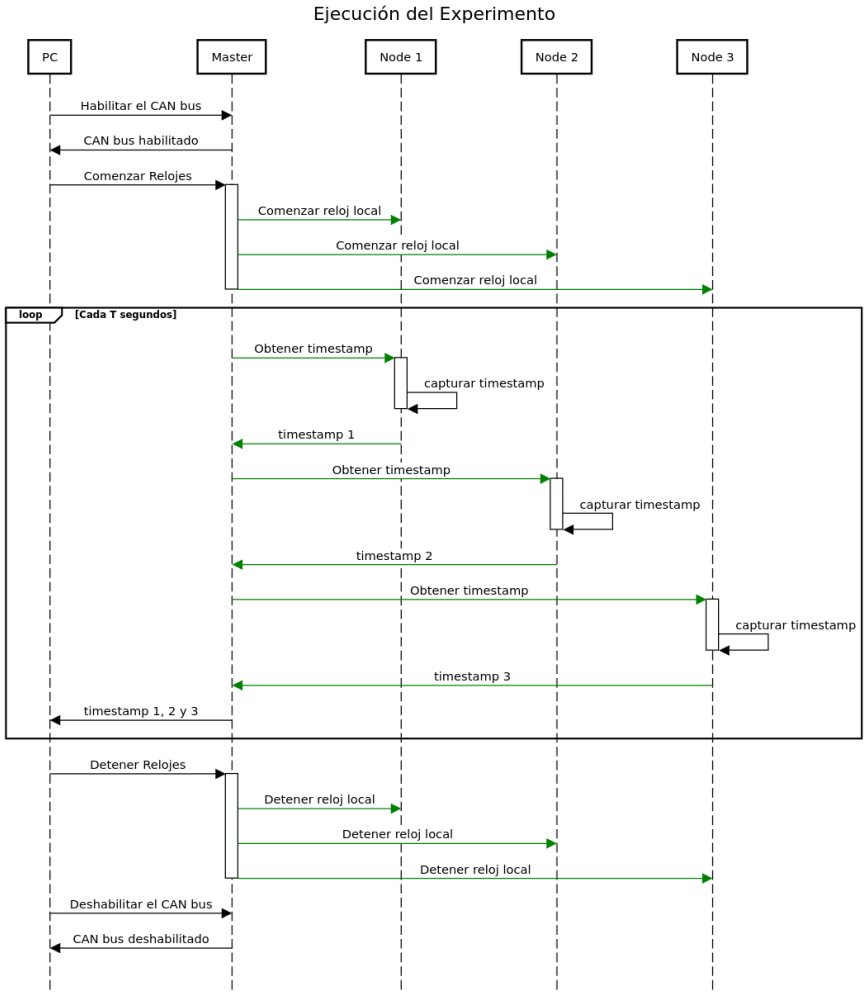

Figura 3: Diagrama secuencial del experimento. Se muestra el flujo de comunicación entre la PC, el Master y los nodos.

Algo que no se destaca en este diagrama, es que la toma del timestamp por parte de los nodos, se hace en la misma interrupción de nuevo mensaje CAN.

4.2. Estructura de los Mensajes de CAN

La trama de CAN tiene un campo llamado ID el cual sirve para informar cuál es el contenido del campo de datos de la trama [2]. Este campo también se usa para definir el nivel de prioridad de la trama. En este sistema hay nodos que son redundantes y que envían información con el mismo contenido. Para que el bus CAN tenga un funcionamiento correcto, no debe permitirse que dos nodos inyecten en el bus mensajes que tengan el mismo ID. Para solucionar esto, se decidió realizar una modificación al uso del campo ID, para que no solo contenga un valor que refiera al contenido del mensaje, sino que además informe a los demás de qué nodo proviene. Esto se muestra a continuación:

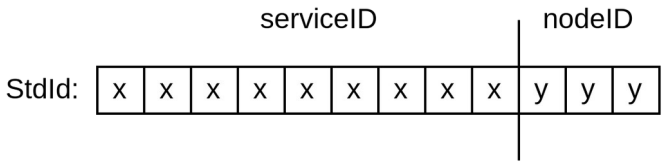

Figura 4: Estructura adopatada para el campo ID de las tramas de CAN.

De esta forma, el sub-campo serviceID es el que informa acerca del contenido del mensaje, mientras que nodeID dice de qué nodo proviene el mensaje. Para la cantidad de bits asignada, se permiten 8 nodos distintos.

Entonces por ejemplo, si un nodo con el nodeID “0” envía el mensaje con el serviceID “timeStampServiceStart” (0x05), luego el StdId del mensaje enviado será:

- StdId(timeStampServiceStart, nodeID) = 0x05<<3 | nodeID = 00000101000

Se muestran otros ejemplos de StdId:

- StdId(timeStampServiceStop, 0) = 0x06<<3 | 0 = 00000110000
- StdId(timeStampServiceData, 1) = 0x07<<3 | 1 = 00000111001
- StdId(timeStampServiceData, 2) = 0x07<<3 | 2 = 00000111010
- StdId(timeStampServiceData, 3) = 0x07<<3 | 3 = 00000111011

De esta manera, hasta 8 nodos distintos pueden enviar mensajes que tengan el mismo contenido y podrá diferenciarse de qué nodo proviene el mensaje.

5. Resultados

En esta sección, se muestran resultados obtenidos a partir de los datos de timestamp recolectados durante 30 minutos. El master solicitó un nuevo dato cada 1 segundo, por lo que se obtuvieron 1800 datos de timestamp por cada placa.

5.1. Drift de Cada Placa

A partir de los datos reportados por cada placa, se midió la diferencia de tiempos entre timestamps consecutivos. Esto se hizo en forma de histograma. En las figuras 5, 6 y 7 se muestran los resultados para la placa 1, 2 y 3 respectivamente. Los tres histogramas se muestran superpuestos en la figura 8.

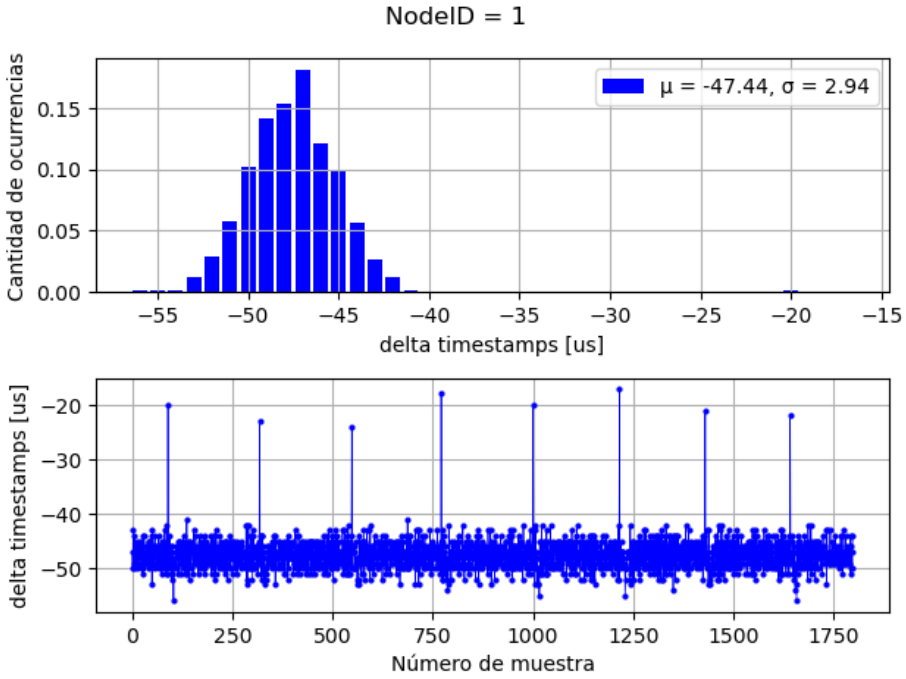

Figura 5: Se muestra un histograma de las diferencias de tiempos, reportadas por la placa 1. A su vez se muestra su variación a lo largo del tiempo. En el histograma, los datos del eje horizontal corresponden a la diferencia respecto de 1 segundo.

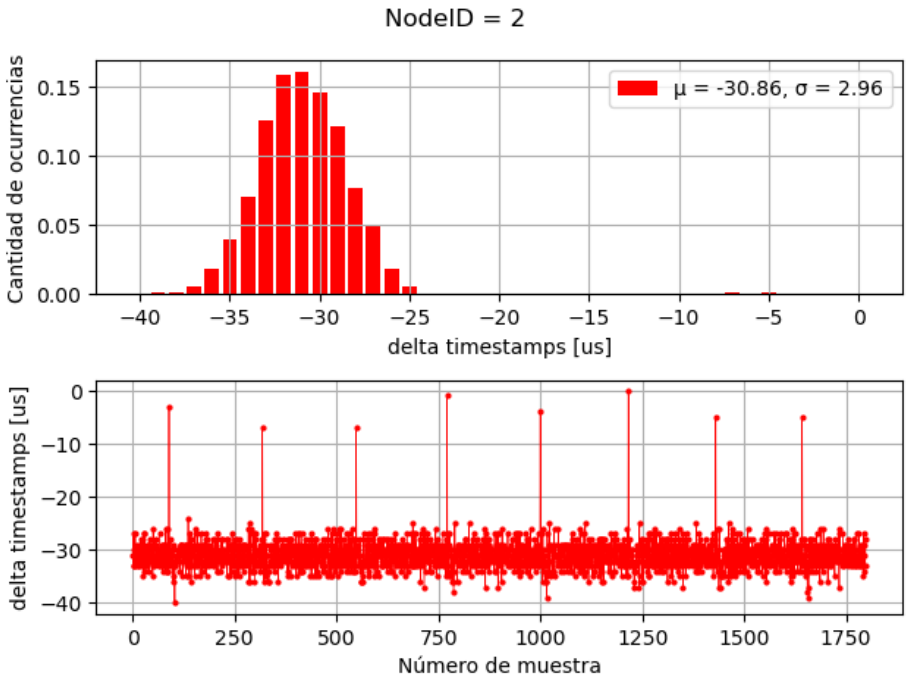

Figura 6: Se muestra un histograma de las diferencias de tiempos, reportadas por la placa 2. A su vez se muestra su variación a lo largo del tiempo. En el histograma, los datos del eje horizontal corresponden a la diferencia respecto de 1 segundo.

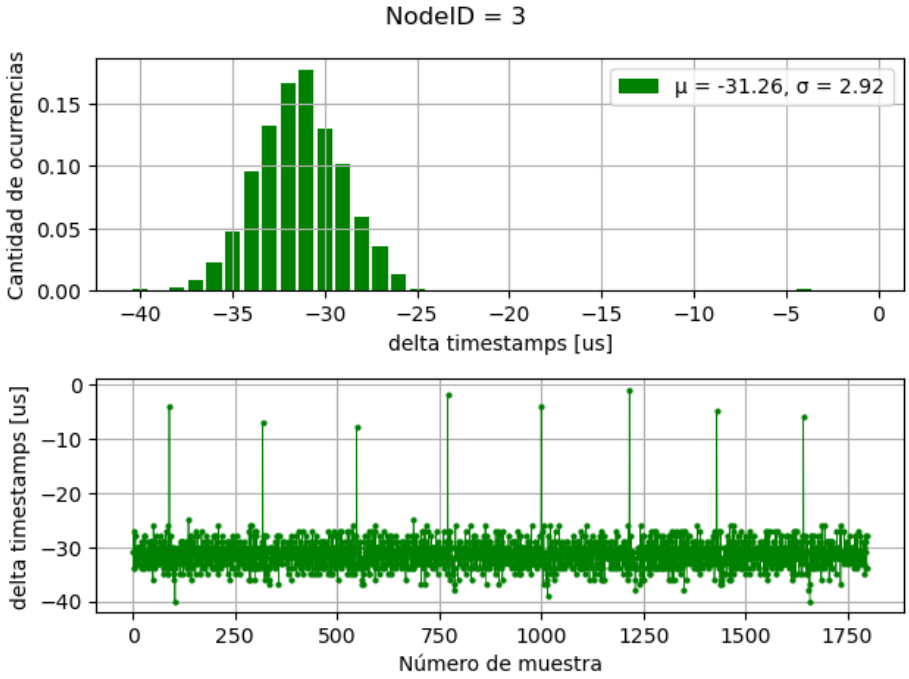

Figura 7: Se muestra un histograma de las diferencias de tiempos, reportadas por la placa 3. A su vez se muestra su variación a lo largo del tiempo. En el histograma, los datos del eje horizontal corresponden a la diferencia respecto de 1 segundo.

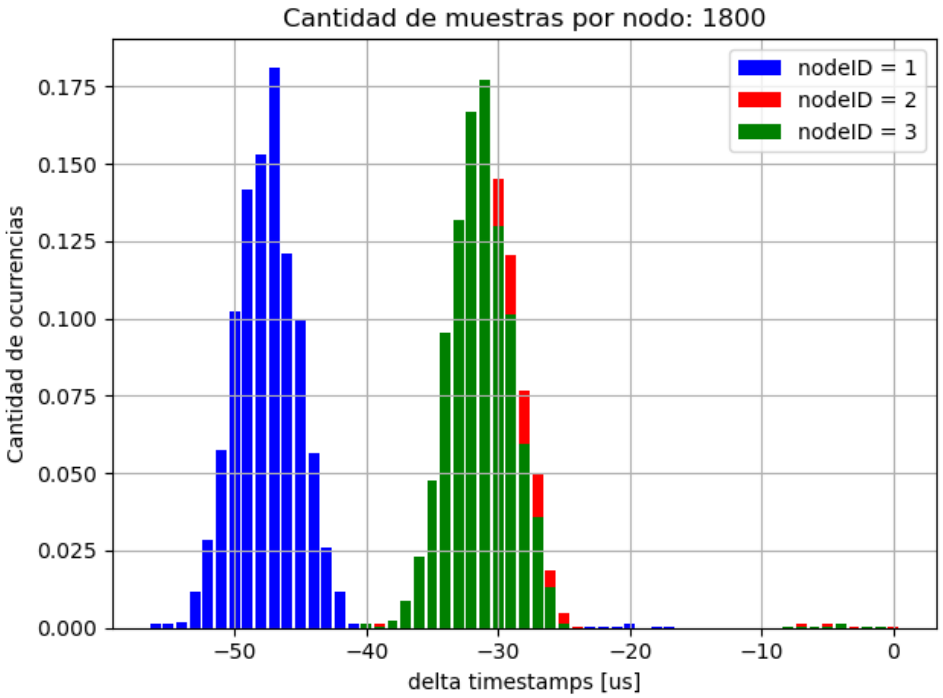

Figura 8: Se muestran los histogramas de las 3 placas superpuestos. Se nota una diferencia de la placa 1, respecto de las 2 y 3.

En los gráficos temporales de las figuras 5, 6 y 7, se muestran picos en las mediciones de las diferencias de timestamps. Estos picos se repiten para las 3 placas, además de repetirse de manera periódica aproximadamente cada 222 segundos. No se hizo un análisis para entender el motivo de estos picos, pero se asume que pueden ser causados por el master, ya que estos se hacen presentes en las 3 placas. Debido a que ocurren al mismo tiempo, estos picos no impactan en el análisis de diferencia de timestamps entre placas.

5.2. Offset Entre Placas

A partir de los resultados de la sección anterior, puede verse que hay una diferencia entre la placa 1 y las placas 2 y 3. En esta sección se grafica el error en los timestamps entre placas, correspondientes al mismo número de muestra:

$$offset_i^{j,k} = |t_i^k - t_i^j|$$

Esta ecuación quiere decir que el offset entre los relojes *j* y *k*, en el instante *i*, es la diferencia de los timestamps tomados en el instante *i*, por los relojes *j* y *k*. En la figura 9, se muestra un gráfico de los offset entre las placas 1 y 2, 2 y 3 y 1 y 3.

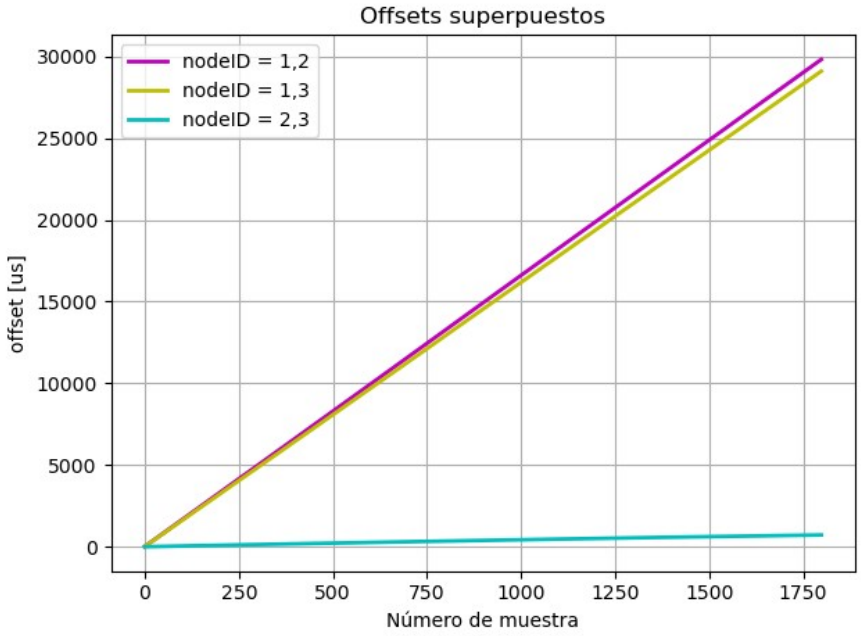

Figura 9: Se muestran los gráficos del offset de las 3 placas superpuestos.

En concordancia con el gráfico de la figura 8, el offset es notablemente mayor cuando se lo mide respecto de la placa 1. La curva magenta, correspondiente al offset entre las placas 1 y 2, alcanza un valor máximo de 29818 us. La curva amarilla alcanza un valor máximo de 29100 us. La curva celeste alcanza un valor de 718 us, un valor que se encuentra muy por debajo de las otras dos.

6. Conclusiones

A partir de los gráficos de histograma se pudo evidenciar que hay un drift distinto en la placa 1 respecto de las 2 y 3. A pesar de esto, en 30 minutos de mediciones, solamente se observó un offset máximo de la placa 1 respecto a las 2 y 3, de 30 ms aproximadamente. Un offset de 30 ms en 30 minutos parece un valor muy bajo, aunque debe analizarse dependiendo la necesidad de la aplicación.

Los resultados obtenidos son válidos para el conjunto de placas slave. En un futuro, si se quisieran agregar otros nodos al bus, nada asegura que sus clocks no tengan un drift tal que requiera utilizarse una sincronización periódica.

7. Referencias

[1] Lamport, L., Shostak, R., & Pease, M. (2019). The Byzantine generals problem. In *Concurrency: the works of leslie lamport* (pp. 203-226).

[2] CAN Specification. «Bosch». En: Robert Bosch GmbH, Postfach 50 (1991), pág. 75.