UNIVERSIDAD AUTÓNOMA DE QUERÉTARO & CONTINENTAL Co.

COMUNICACIÓN SCI ENTRE DOS TARJETAS DELFINO UTILIZANDO CONTROL DE VERSIONES

Programa de educación DUAL Continental-UAQ

Semestre: 2020B

TUTORES:

M. Alejandro Rivera Garay

Dr. Mariano Garduño Aparicio

|  |
| --- |
| ALUMNOS:  Iván Eduardo Andrade Jains  Andrea Abigail Anievas Guerrero  Farid Iván Arriaga Tejeda  Luis Ángel Chávez Espínola  Rubén Alejandro García Sánchez  Jesús Alberto Herrera Curiel  Claudia Beatriz Reséndiz Jurado  Gabriela Suárez Páez  Mayra Denisse Uribe Escobar |

INTRODUCCIÓN

La comunicación serie o comunicación secuencial, es el proceso de envío de datos de un bit a la vez, de forma secuencial, sobre un canal de comunicación o un bus.

Las características más importantes de la comunicación son:

* Velocidad de transmisión
* Bits de datos
* Bits de parada
* Bit de paridad

Para que dos puertos puedan comunicar es necesario que las características sean iguales. La velocidad de transmisión indica el número de bytes por segundo que se transfieren y se miden en baudios(bauds). [1]

Ancii es una herramienta para comunicarse por medio de comunicación serial, existe ancii estándar(0 a 127) es decir utiliza 7 bits y ancii extendido(0 a 255) por lo que utiliza 8 bits, un paquete se refiere a una transferencia de base Incluyendo los bits de inicio y parada así como los datos bit de datos debido a que el número actual de bits depende en el protocolo que se selecciona el término paquete se usa para referirse a todos estos casos. [2]

Además, me comunicación serial puede ser utilizada para adquisición de datos si se usa un conjunto con un dispositivo remoto de muestreo el puerto serial recibe y envía datos envía bit de información un bit a la vez aún y cuando esto es más lento Que la comunicación en paralelo que permiten la transmisión de un by completo por este método de comunicación es más sencillo y puede alcanzar mayores distancias.

GIT

Git es un software de control de cambios en diferentes versiones de un código. Git permite realizar un seguimiento de los cambios en archivos y, si es necesario, restaurar las versiones anteriores. El uso de Git permite a varias personas trabajar simultáneamente en el mismo código sin tener que sobrescribir los cambios del otro.

Además, Git permite la creación de múltiples alternativas. Estas versiones se pueden utilizar para probar nuevos plugins y estructuras de sitios web o para el desarrollo independiente de varios scripts PHP.

Interfaz de Comunicación Serial de la tarjeta F28377S

El módulo de la Interfaz de Comunicación Serial, o por sus siglas en ingles SCI (Serial Communication Interface), es un puerto de Entradas y Salidas en serie que hace posible la comunicación asincrónica entre la tarjeta F28377S y otros dispositivos periféricos. Usualmente se le conoce por sus siglas en inglés como UART (Universal Asynchronous Receiver Transmitter) y es utilizado comúnmente de acuerdo al estándar de comunicación RS232.

El transmisor y receptor del SCI cuentan con un FIFO de 16 palabras, cada uno con sus propios bits de habilitación e interrupción. Ambos pueden ser operados de forma independiente para comunicaciones Half-Duplex, o de forma simultánea para comunicaciones Full-Duplex. La tasa de bits es programable para diferentes velocidades de comunicación a través de un registro de 16 bits.

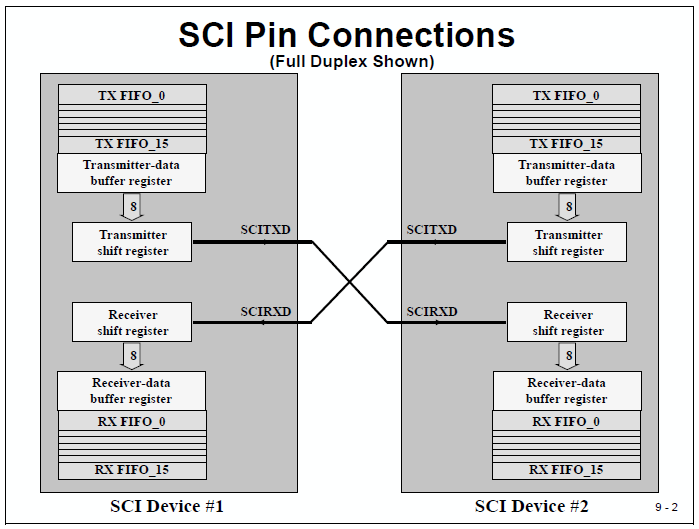


Figura 1.- Diagrama de composición del SCI para una comunicación Full Duplex

OBJETIVO

Con el entendimiento del funcionamiento del código para comunicación SCI, realizar la conexión y la comunicación entre dos tarjetas Delfino tomando en consideración los requisitos expuestos por el profesor utilizando Git como principal herramienta para control de versiones.

MATERIAL

A continuación, se enlista el material utilizado para la realización de la práctica:

* 2 tarjetas Delfino LaunchPad – LaunchXL-F28377S
* 3 cables DuPont Hembra Hembra

**DESARROLLO**

Para el desarrollo de esta práctica se empleó parte del código que hicimos anteriormente para que entre dos tarjetas se comunicaran y formaran la palabra “Hola”. Como en esta ocasión no se requería únicamente de caracteres, se optó por utilizar un número concreto de strings en el vector msg, en cuánto a las funciones se emplearon las mismas con las que ya habíamos venido trabajando, sin embargo, se sumó una llamada “scic\_rcv\_msg”, tal y como se puede observar en la siguiente imagen.

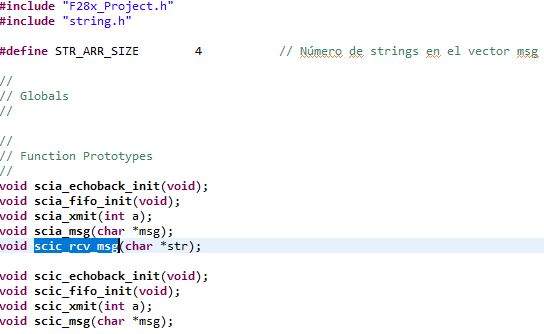


Figura 2. Declaración de funciones.

Posteriormente llegamos al main del programa, en esta parte declaramos dos contadores, uno llamado i que es empleado para avanzar en el array msg y con el contador it que es el encargado de que el programa se ejecute únicamente 10 ocasiones, también se declara el string que será usado para almacenar lo que se recibe a través del puerto SCIC y está el vector que contiene las palabras con las que se compararán los mensajes recibidos y los mensajes que deben enviarse, adicionalmente se inicializan los puertos de comunicación SCIA y SCIC, sin embargo no se dará una explicación a profundidad de dichas cuestiones puesto que en el código anterior fueron utilizadas de igual forma.

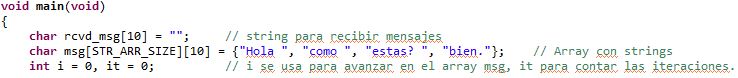


Figura 3. Declaración de contadores y variables tipo char.

Dentro del main también se encuentra la lógica principal del programa, donde únicamente se ejecuta 10 veces y al término de estas 10 veces se envía el mensaje “Fin”, dentro del main se hace llamar a la función scic\_rcv\_msg y a su vez obtener el valor que contendrá la variable rcvd\_msg. Posteriormente se hará una explicación detallada de dicha función. Tras haber obtenido el valor de la variable rcvd\_msg se realiza una comparación con la variable msg que contiene las palabras con las que se va a formar la oración y se selecciona la posición del vector mediante el contador i que dependiendo de la iteración en la que nos encontremos irá cambiando y esperará la palabra correspondiente a la iteración en la que se encuentra. Si la palabra que se recibe es igual a la palabra en el vector el comando strcmp arroja un cero, en caso de que sean iguales se procede a imprimir por el SCIA en la consola del CCS la palabra recibida, se da un delay de 2 segundos según lo establecido por el Dr. Mariano y se procede a enviar la palabra del vector en la posición correspondiente, en el momento en que el contador i supera el valor de la cantidad de palabras almacenadas en el vector, se reinicia a 0 y a su vez se aumenta en uno el contador it. Una vez finiquitadas las 10 iteaciones, se sale del ciclo while y se envía por el puerto SCIA la palabra Fin.

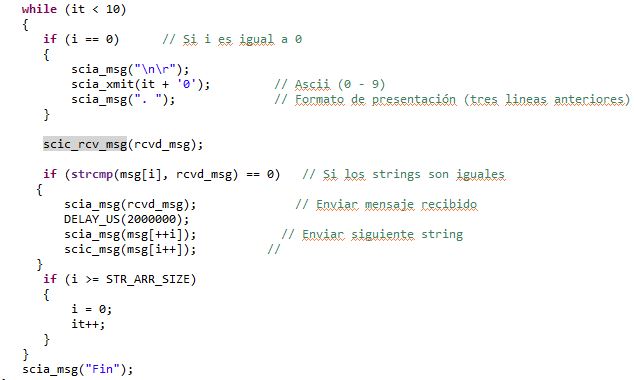


Figura 4. Estructura lógica principal del programa.

Por último, tenemos la función scic\_rcv\_msg, la cual se encarga de recibir la trama de datos que forma la palabra que se ha enviado a través de la otra tarjeta. Su funcionamiento se explica en la siguiente figura.

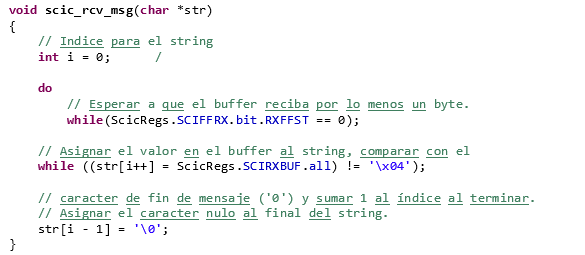


Figura 5. Código de la función para recibir strings.

El resto del código está constituido por las funciones previamente empleadas y estudiadas, como lo son las funciones para configurar la velocidad de comunicación y la forma en la que va a estar constituida la trama de datos. Solo queda agregar que se hizo una pequeña modificación a la función scic\_msg la cual es la encargada de transmitir la trama de datos a través del puerto SCIC, se hizo que tras enviar el mensaje enviara también el valor hexadecimal 04, que equivale a un 4 en decimal y si se observa la tabla ASCII se podrá constatar que es el carácter que indica el fin de la transmisión, carácter que es empleado en la función scic\_rcv\_msg para encontrar el final de la trama de datos.

En caso de requerir mas detalle sobre los códigos favor de consultar los archivos anexos correspondientes a los códigos.

Conexiones Realizadas

Utilizando la documentación de la tarjeta se determinó la posición de los pines correspondientes al módulo SCIC para realizar la interconexión de las tarjetas.

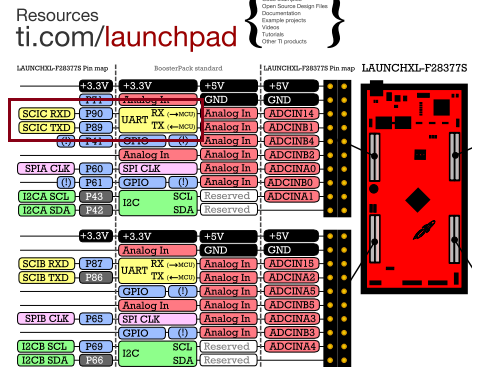


Figura 6.- Los pines 3 y 4 de J1 de la tarjeta son los correspondientes al módulo SCIC

Se conectaron las tierras de las tarjetas y el Tx de la tarjeta A con el Rx de la Tarjeta B y así mismo el Rx de la tarjeta A con el Tx de la tarjeta B.

Una vez que se realizó la conexión física, se procedió a cargar los códigos correspondientes a cada tarjeta. Ya con el código cargado en cada tarjeta, se utilizó la terminal de Code Composer para observar los resultados obtenidos y así visualizar la información que estaba transmitiéndose y recibiéndose en cada una de las tarjetas.

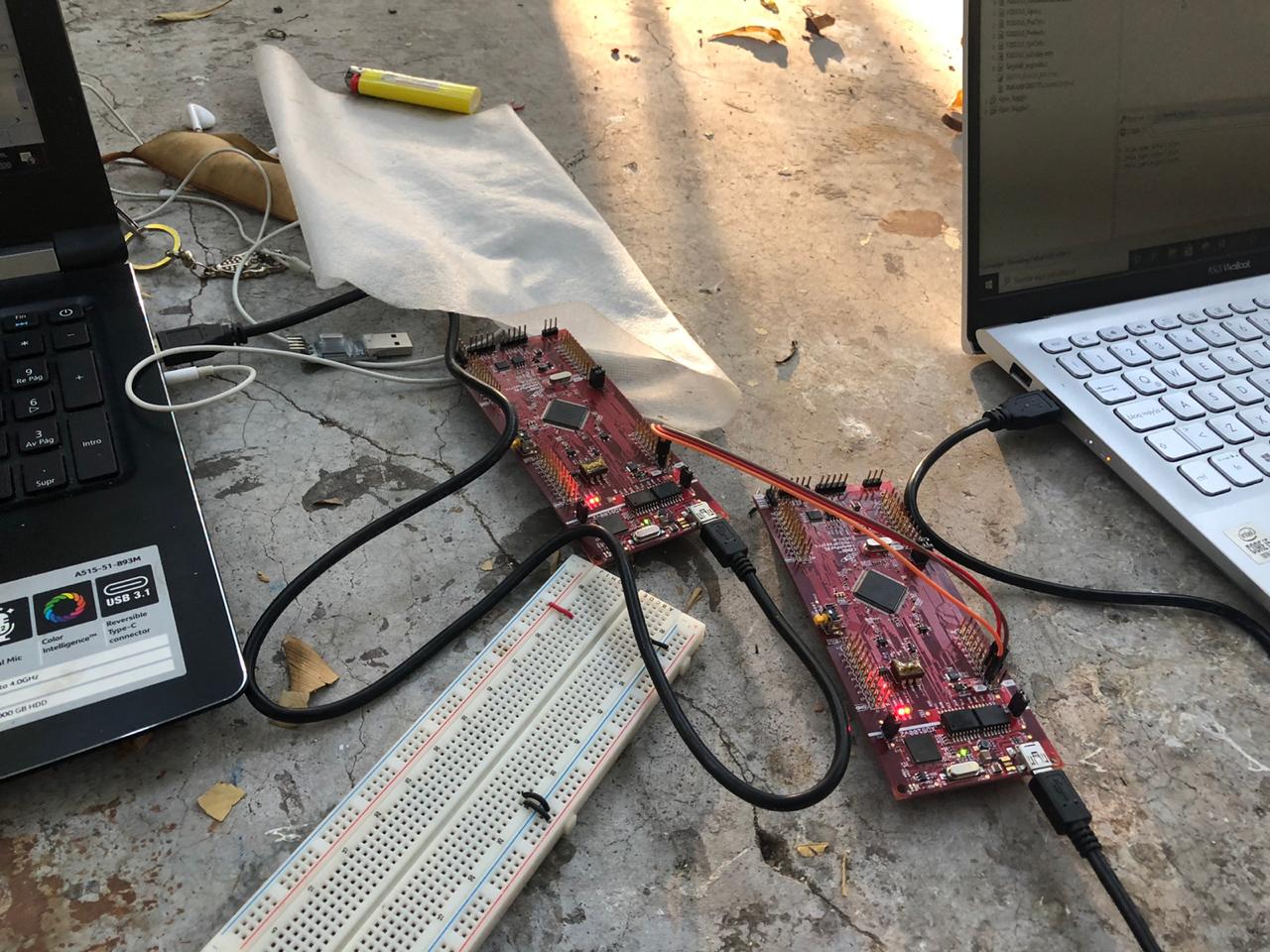


Figura 7.- Conexión del puerto SCIC en ambas tarjetas

Como se observa en las figuras siguientes, los resultados fueron los esperados y la comunicación entre las tarjetas fue exitosa.

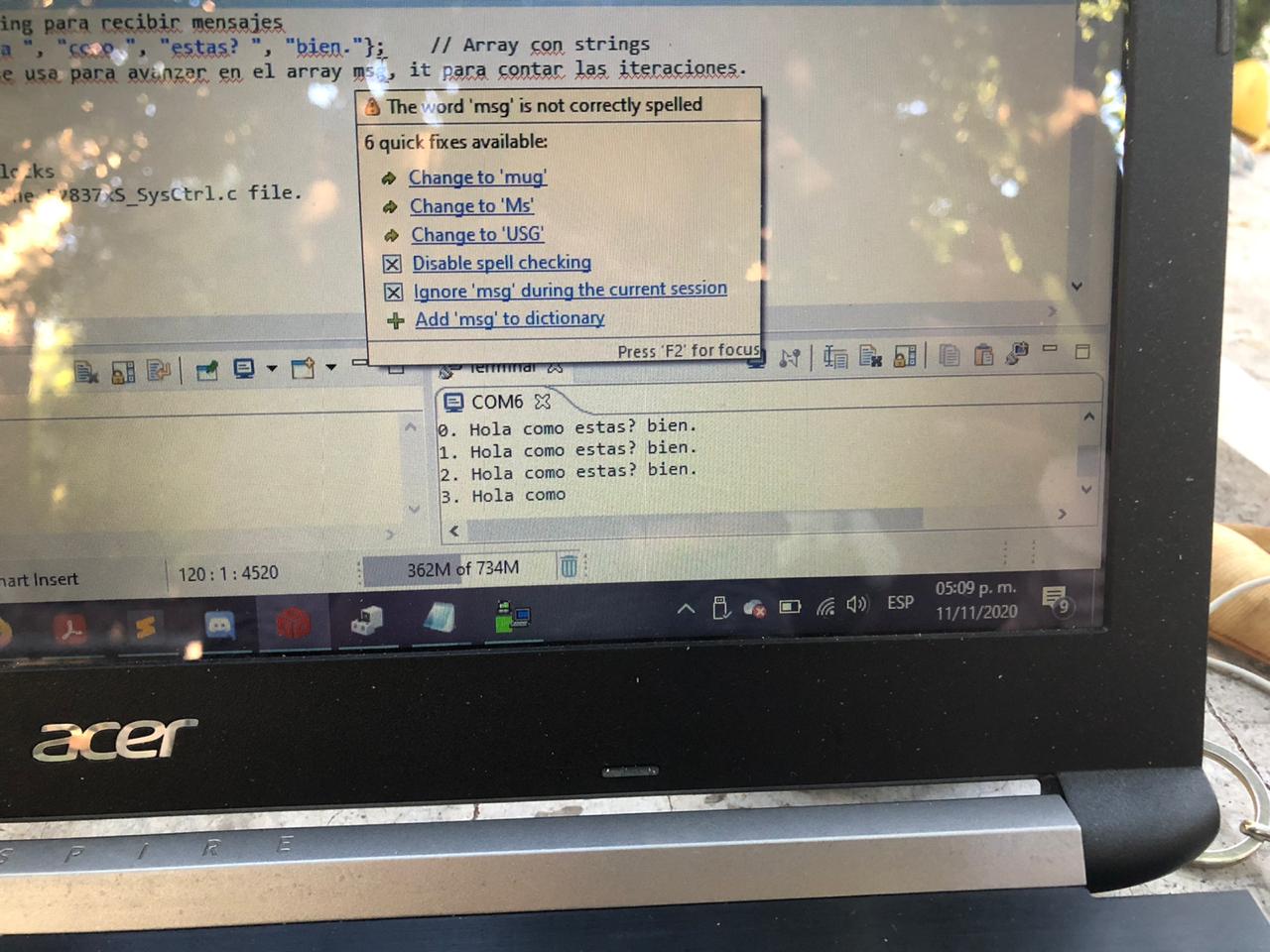


Figura 8.- Terminal serial de Code Composer, desplegando la información enviada y recibida en la Tarjeta A.

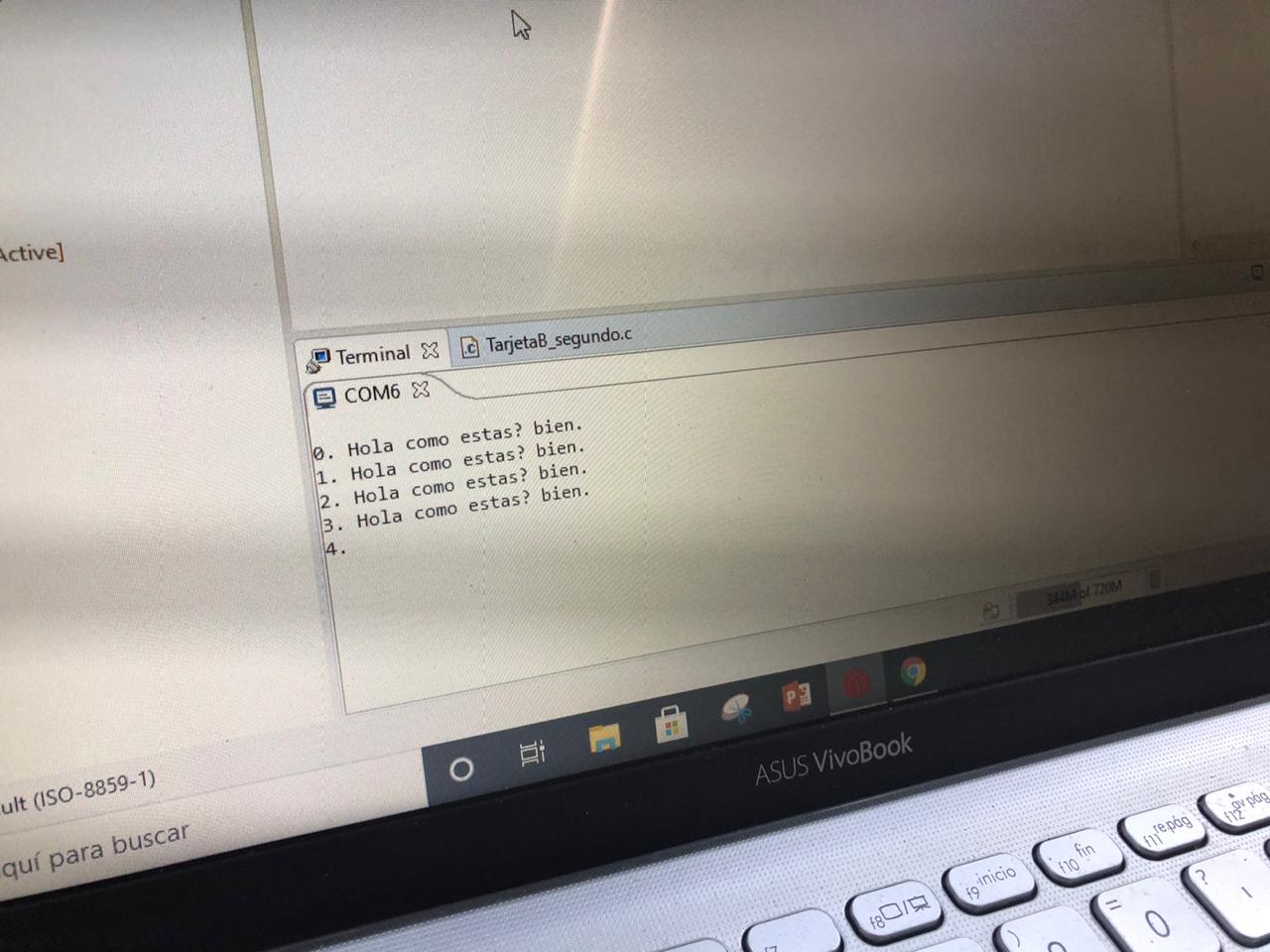


Figura 9.- Terminal serial de Code Composer, desplegando la información enviada y recibida en la Tarjeta B.

A continuación, se muestran los network graph de GitHub correspondientes al repositorio del código y al repositorio de la práctica.

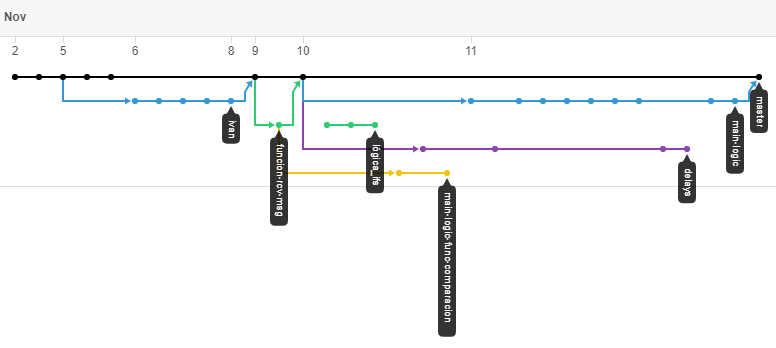


Figura 10.- Network Graph correspondiente al repositorio del código.

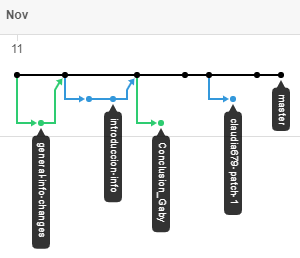


Figura 11.- Network Graph correspondiente al repositorio del reporte

El video de los resultados se encuentra en los anexos a este reporte.

CONCLUSIONES

Iván.- Con el desarrollo de esta práctica nos hemos familiarizado con el uso de Git y GitHub para poder trabajar sobre un mismo código, tomando como base el código con el cual ya habíamos trabajado con anterioridad, para cumplir con los requisitos del programa que habían sido establecidos por el profesor. En esta ocasión no solo comunicamos un solo carácter de tarjeta a tarjeta si no que ahora transmitimos palabras o cadenas de caracteres para realizar dicho funcionamiento se realizaron varias lógicas sobre todo para el almacenamiento de la cadena recibida y la comparación de la misma con el mensaje esperado. Primeramente, para el almacenamiento de la cadena recibida se había implementado una lógica dentro del ciclo infinito del programa, sin embargo, esta se sintetizo en una función. Para la comparación se realizaron dos lógicas una utilizando una función definida desde cero y otra donde se utilizaba la función ‘strcmp’ de la librería ‘string.h’. Finalmente se optó por usar la implementación de la función existente en la librería. Todos estos ajustes han sido identificados y ejecutados con facilidad gracias a la implementación de un control de versiones por Git, por lo que será muy indispensable adentrarnos al uso de este tipo de plataformas a fin de que el desarrollo de un proyecto como este sea llevado a cabo una forma organizada y adecuada.

Andrea.- En esta práctica tuvimos dos desafíos que consistían en recibir frases en vez de letras, en donde el mayor reto era la comparación de las frases al recibirse, esto se arregló al definir los string e implementar un while donde se realiza esta comparación. El siguiente reto que presentamos fue organizarnos y realizar el reporte ya que teníamos que utilizar git para modificar cierto documento.

Farid.- Considero que la práctica fue la introducción al trabajo colaborativo con el uso del manejador de versiones con alojamiento en la nube(GitHub). Fue necesaria la constante comunicación entre los miembros de equipo, la coordinación en el equipo fue muy buena. Puedo destacar que el amplio conocimiento de ciertos miembros del equipo en temas específicos permite un avance constante y si bien fue necesaria la revisión de varias versiones de código para llegar al resultado final puedo decir que trabajamos correctamente.

Personalmente me gustaría fortalecer mis habilidades en el desarrollo de software en el área que se encuentran las practicas que hemos estado realizando.

Luis.- El reto en esta ocasión fue el hecho de que no solo teníamos que mandar caracteres, sino que teníamos que controlar palabras para formar frases, un cierto número de veces e indicar cuando el proceso terminaba. El controlar el número de veces que se ejecutaba el código no fue especialmente complejo, así como tampoco lo fue el indicar que el proceso había terminado, lo más complejo a mi parecer fue el formar las oraciones de forma que fueran entendibles para el usuario que las visualizaría en la consola de la hiperterminal, asimismo fue complejo el comparar las palabras para poder determinar qué palabra era la que debía enviarse, para hacer esto se empleó la función "strcmp", en dicha función se comparan dos variables de tipo char y si son iguales el resultado será cero, una vez comprobado que la palabra que se recibió fue la esperada, se procede a enviar la palabra correspondiente y así por 10 veces, empezando el contador en 0. Una vez finalizadas las 10 cuentas el programa muestra la palabra Fin en ambas tarjetas. Cabe destacar que de nueva cuenta, pese haber tenido un delay inicial de 3 segundos para que iniciara la comunicación entre tarjetas, de nueva cuenta se tuvo que hacer uso del botón pausa y de reinicio para que el programa se ejecutara de manera correcta.

Rubén.- A diferencia de la práctica anterior, esta vez teníamos que enviar y comprobar el envío de cadenas completas de caracteres, en igualar de caracteres aislados, lo que no supuso un reto mayor aunque sí fue necesario hacer una correcta coordinación de los códigos para ambas tarjetas a cada parte del algoritmo. El trabajo del equipo se repartió de una manera mucho más organizada.

Jesús.- El valor de esta práctica se encuentra en la recepción de un arreglo de caracteres, ya que, a diferencia de la práctica anterior en la cual solo debíamos comunicarnos por medio de un byte cada vez, en esta ocasión debíamos reconocer una frase completa y fue más sencillo trabajar con un carácter que indicara el fin de la transmisión el cual por suerte ya tiene su representación en el código ASCII (0x04). De esta forma se logró entablar la comunicación entre ambas tarjetas sin problema alguno.

Claudia.- En esta práctica lo que se quiso reforzar fueron los conocimientos en el GitHub ,aprender a usarlo y saber cómo se puede aplicar con Code Composer y con Word para la elaboración del reporte asignado mientras que usamos GitHub en code composer para poder hacer las ramas e ir modificando el código base de echoback este añadiéndole un poco más de complejidad ya que ahora trabajaríamos con cadenas de caracteres y no solo con un carácter.

Gabriela.- Esta práctica fue un poco más elaborada que la anterior, sin embargo, al ser prácticamente el mismo funcionamiento que la práctica pasada, en términos de código no hubo muchos cambios y pudimos realizar el código más fácilmente. Creo que más que nada, esta práctica nos permitió familiarizarnos con el uso de GIT para control de versiones y nos ayudó a trabajar un poco más en conjunto como equipo. Creo que este conocimiento sobre la plataforma de GIT es muy valioso y nos va a servir a todos en nuestro futuro profesional y para nuestro desarrollo personal, así como esta experiencia de trabajo en equipo, que también es importante desarrollarla en el ámbito académico.

Mayra.- Esta práctica igualmente se hizo usando de base el código de echoback, sin embargo se hicieron algunas modificaciones en el código para ajustarlo y también para que el mensaje enviado por las tarjetas se mostrará de forma más limpia en las terminales. Al momento de conectar las tarjetas nos encontramos con el mismo problema de que no se comunicaban sino hasta hacer un reinicio de cada una de ellas antes de correr el código. Después de hacer ese reinicio ya se pudieron enviar y mostrar los mensajes requeridos.

BIBLIOGRAFÍA

**[1]** Catthoor, F., Danckart, K., Kulkarni, C.. (2002). Data access and storage management for embedded programmable processors. Boston: Kluwer Academic.

**[2]** Heslop K., Karst J., Prensky S., Schmitt D.. (2000). LAS 3.0. Log ASCII Standard. .: Calgary, Alberta.