Instituto Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

EL3313 Taller de Diseño Digital

Cuestionario previo laboratorio 2:

Lógica secuencial

Profesor:

M.Sc. Kaleb Alfaro Badilla

Realizado por:

Brayan de Jesús Barquero Madrigal

Keylor David Muñoz Soto

Jeffrey Isaac Salas Quiel

6 de septiembre del 2024

ÍNDICE

1. PREGUNTA 1

Investigue sobre el funcionamiento de máquinas de estado finitos. Explique la diferencia entre una máquina de Moore y una de Mealy, y muestre la diferencia por medio de diagramas de estados y señales.

Las máquinas de estados finitos (FSM) son circuitos que poseen un k número de registros y poseen 2^k estados únicos, un FSM está constituido por dos bloques combinacionales, *next state logic*, *output logic* y un registro que guarda los estados. Existen dos máquinas de estados finitos, la máquina de Moore y la de Mealy. En una máquina de estados de Moore las salidas dependen únicamente del estado actual en la máquina, mientras que en una máquina de Mealy las salidas dependen del estado y la entrada actual. [1]

En las siguientes figuras se muestran los diagramas de las máquinas de estados de Moore y Mealy respectivamente.

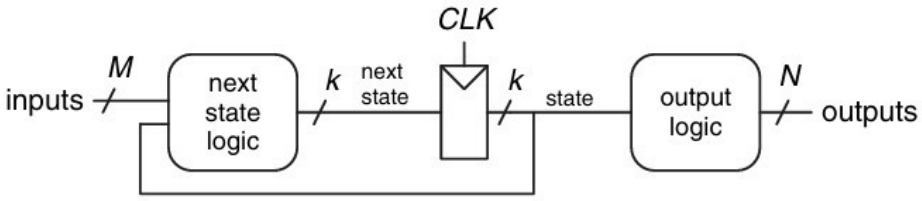

Figura 1.1: Máquina de estados finitos de Moore. [1]

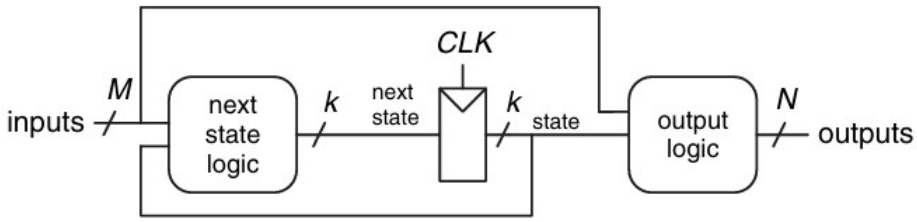

Figura 1.2: Máquina de estados finitos de Mealy. [1]

2. PREGUNTA 2

Explique los conceptos de *setup time* y *hold time*. ¿Qué importancia tienen en el diseño de sistemas digitales?

Para que el circuito detecte las entradas correctamente, estas deben haberse estabilizado al menos un *setup time* (t_{setup}) antes del flanco positivo del reloj y deben permanecer estables al menos un *hold time* (t_{hold}) después del flanco positivo del reloj. El tiempo de apertura (*aperture time*) del circuito es la suma de *setup time* con *hold time* y es el tiempo total que la entrada debe permanecer estable para que sea detectada correctamente. [1]

Estos conceptos son importantes en el diseño de sistemas digitales porque al asegurar que las entradas se detecten correctamente, también se garantiza que las salidas muestren un resultado esperado. [1]

Por ejemplo, si colocan varias veces las mismas entradas al sistema digital probablemente se espere que la salida no varíe, pero si no se toma en cuenta el *setup time* o *hold time* es posible que alguna de las entradas no sea detectada provocando que la salida del sistema sea distinta ocasionalmente.

3. PREGUNTA 3

Explique los conceptos de tiempos de propagación y tiempos de contaminación en circuitos combinacionales. Investigue sobre la ruta crítica y cómo esta afecta en el diseño de sistemas digitales complejos; por ejemplo, un procesador con pipeline. Investigue su relación con la frecuencia máxima de operación de un circuito.

El tiempo de propagación (t_{pd}) es el tiempo máximo desde que la entrada cambia hasta que la salida alcanza su valor final. El tiempo de contaminación (t_{cd}) es el tiempo mínimo desde que la entrada cambia hasta que la salida empieza a cambiar su valor. [1]

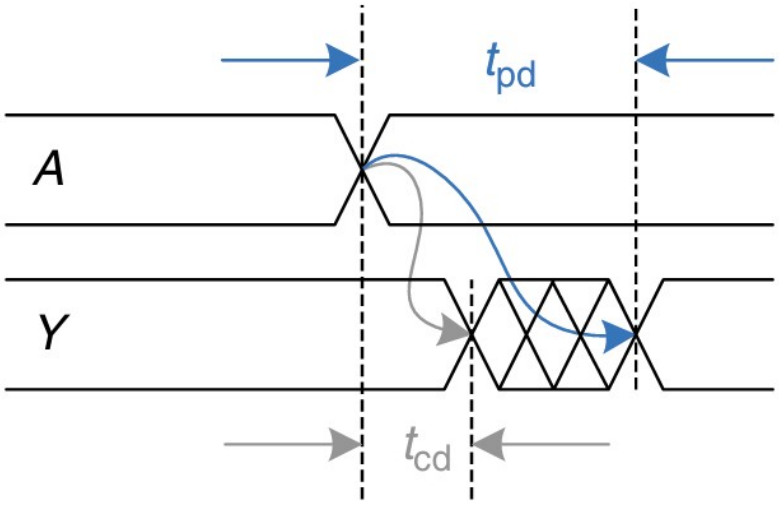

Figura 3.1: Tiempo de propagación y de contaminación. [1]

La ruta crítica es la ruta más larga que hay de una entrada a una salida, la ruta más larga generalmente es la que pasa por el mayor número de compuertas, esta ruta limita la rapidez a la que el circuito o en este caso procesador opera. [1]
Esto significa que el tiempo de propagación total de la ruta crítica es el que define el periodo de trabajo del procesador ya que es la salida que más tiempo tarda en ser procesada, por lo que limita la frecuencia a la que puede trabajar.
El tiempo de propagación total es la suma de los tiempos de propagación en las compuertas de la ruta crítica y el tiempo de contaminación total es la suma de los tiempos de contaminación en las compuertas de la ruta más corta. [1]

4. PREGUNTA 4

Investigue sobre las mejores prácticas para la asignación de relojes y división de frecuencia en FPGAs. En este apartado haga énfasis en el uso de las entradas habilitadoras de reloj (*clock enables*) presentes en las celdas de la FPGA, para lograr tener tiempos de ejecución diferentes a lo largo del sistema mientras se utiliza un solo reloj.

Fpgas se utiliza para la asignación de relojes y la división de frecuencias en sistemas digitales que operan a diferentes velocidades. Las entradas de habilitación de reloj son una forma de controlar la activación de un módulo en momentos designados, lo cual es necesario para regular diferentes frecuencias de operación usando un solo reloj. [2, 4]
El reloj se utiliza para habilitar Esto puede ayudar a ahorrar consumo de energía y mejorar el diseño del reloj en sistemas donde diferentes partes del circuito necesitan diferentes tiempos de ejecución La habilitación del reloj se puede utilizar para sincronizar con un único reloj global, pero algunas partes pueden ser un poco más bajas al activarse solo en ciclos de reloj específicos [3]
Implementación práctica Si un bloque necesita funcionar a la mitad de la frecuencia del reloj principal, se puede generar una señal de habilitación que se activa cada dos ciclos de reloj

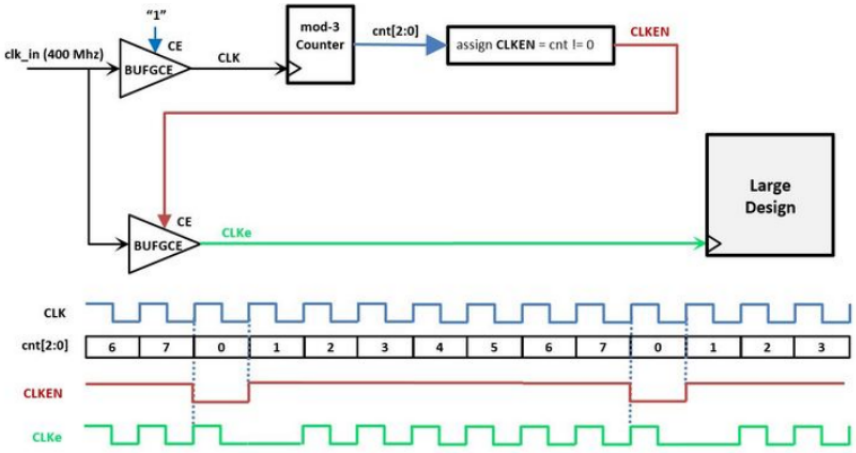

Figura 4.1: Periodo de CLK. [5]

5. PREGUNTA 5

Investigue sobre el fenómeno de rebotes y ruido en pulsadores e interruptores. Defina qué técnicas digitales (circuitos) se utilizan para cancelar este fenómeno. Además, investigue sobre los problemas de metastabilidad cuando se tienen entradas ast'ncronas en circuitos digitales. Finalmente, presente circuitos que permitan la sincronización de entradas como pulsadores e interruptores.

El fenómeno de rebotes en pulsadores e interruptores es causado por transiciones mecánicas imperfectas, que generan

múltiples cambios rápidos entre los estados de encendido y apagado. Esto produce ruido que puede afectar el comportamiento de los sistemas digitales. Para evitar este problema, se utilizan las siguientes técnicas:

- **Filtros RC**: Suavizan las transiciones, eliminando picos de señal indeseados.
- **Circuitos de debouncing basados en flip-flops y contadores**: Estos detectan y estabilizan la señal, asegurando que solo una transición válida sea registrada por la lógica del sistema. [8]

Problemas de Metastabilidad: La metastabilidad ocurre cuando se introducen señales asíncronas, como las provenientes de entradas externas, en circuitos sincronizados por reloj. Si el flip-flop no puede resolver un estado lógico dentro del tiempo requerido, esto puede llevar a un comportamiento indeterminado en el circuito. Para reducir el riesgo de fallos, se utilizan sincronizadores de doble o triple etapa, donde los flip-flops en cascada estabilizan la señal antes de que sea procesada. [7]

Circuito de Debouncing: La siguiente figura muestra el comportamiento de la señal de un pulsador con y sin circuito de debouncing. [6]

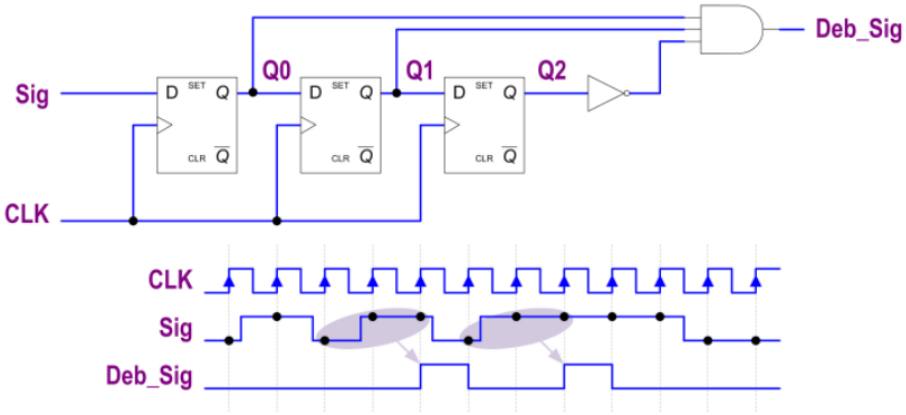

Figura 5.1: circuito de debouncing. [9]

El uso de circuitos de debouncing y sincronizadores de etapas múltiples es crucial para asegurar la correcta interpretación de las señales en sistemas digitales, eliminando el ruido provocado por rebotes y minimizando el riesgo de errores debido a la metastabilidad.

6. PREGUNTA 6

Investigue sobre la especificación de la interfaz SPI. Preste atención a los aspectos necesarios para poder diseñar un controlador maestro de SPI, además de los diferentes modos de SPI.

El *Serial Peripheral Interface (SPI)* es una interfaz de comunicación síncrona y full-dúplex ampliamente utilizada entre microcontroladores y circuitos integrados periféricos, como sensores, ADCs, DACs, registros de desplazamiento, SRAM,

entre otros. Esta interfaz permite que tanto el dispositivo maestro (*main*) como el esclavo (*subnode*) transmitan datos simultáneamente. [10]

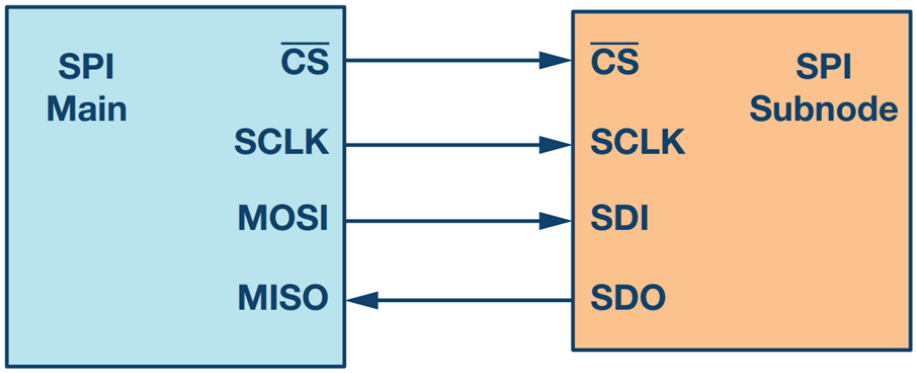

Figura 6.1: Disposición del SPI con maestro y un esclavo. [10]

La interfaz SPI generalmente está compuesta por las siguientes cuatro líneas [10]:

- ***Clock* (SPI CLK o SCLK):** Señal de reloj generada por el maestro. Esta señal sincroniza la transmisión de datos entre el maestro y el esclavo.
- ***Chip Select* (CS):** Es una señal de selección de chip, utilizada por el maestro para habilitar o seleccionar un esclavo específico. Esta señal es típicamente activa en bajo, lo que significa que el maestro debe enviar un "0"lógico para activar el esclavo.
- ***Main Out, Subnode In* (MOSI):** Es la línea de datos por la que el maestro envía datos al esclavo.
- ***Main In, Subnode Out* (MISO):** Es la línea de datos por la que el esclavo envía datos al maestro.

Según [10], la interfaz SPI tiene cuatro modos de operación, definidos por dos parámetros: CPOL (*Clock Polarity*) y CPHA (*Clock Phase*). Estos parámetros determinan la polaridad del reloj en estado inactivo y el momento en el que los datos son muestreados o desplazados. A continuación, se describen los modos:

- **Modo 0 (CPOL = 0, CPHA = 0):** Reloj en nivel bajo en estado inactivo. Los datos se muestrean en el flanco de subida del reloj y se desplazan en el flanco de bajada.
- **Modo 1 (CPOL = 0, CPHA = 1):** Reloj en nivel bajo en estado inactivo. Los datos se muestrean en el flanco de bajada del reloj y se desplazan en el flanco de subida.
- **Modo 2 (CPOL = 1, CPHA = 0):** Reloj en nivel alto en estado inactivo. Los datos se muestrean en el flanco de bajada del reloj y se desplazan en el flanco de subida.
- **Modo 3 (CPOL = 1, CPHA = 1):** Reloj en nivel alto en estado inactivo. Los datos se muestrean en el flanco de subida del reloj y se desplazan en el flanco de bajada.

Esto se resume en la siguiente figura:

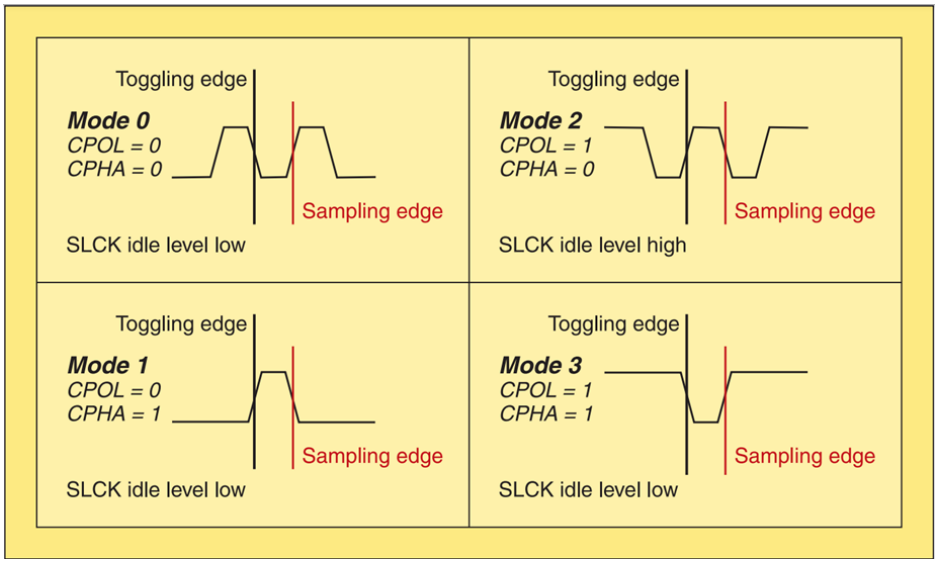

Figura 6.2: Modos de operación de la interfaz SPI. [11]

7. PREGUNTA 7

Investigue sobre la comunicación serie UART. Preste atención a las diferentes características de configuración necesarias para la comunicación serie mediante UART (por ejemplo, *baud rate*, paridad, etc). Además, investigue cómo puede utilizar puertos seriales en su computadora, considerando el sistema operativo que utilice.

UART es un protocolo de comunicación en serie que se utiliza ampliamente para la comunicación entre dispositivos electrónicos como microcontroladores, sistemas embebidos y computadoras. UART utiliza dos cables, uno para la transmisión de datos (Tx) y otro para la recepción de datos (Rx). [12]

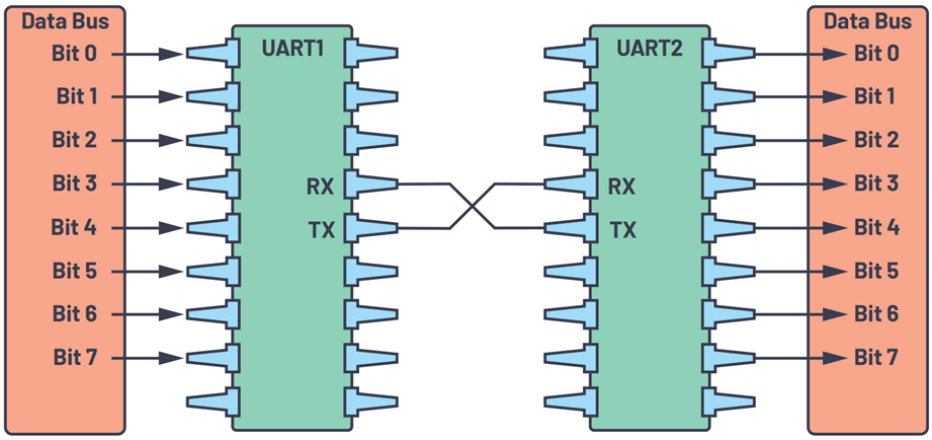

Figura 7.1: Disposición comunicación serial UART. [12]

UART realiza una comunicación asíncrona, lo que significa que no utiliza una señal de reloj para sincronizar el envío de

bits desde el dispositivo transmisor al receptor, en cambio, tanto el transmisor como el receptor deben estar configurados con la misma tasa de baudios para garantizar una transmisión y recepción correcta de los datos. La tasa de baudios o *baud rate* es la cantidad de bits transmitidos por segundo. Algunas tasas comunes son 9600, 19200, 115200, hasta 1500000 baudios. [12]

El *data frame* puede ser de 5 a 8 bits de longitud (o hasta 9 bits si no se utiliza un bit de paridad). El bit de paridad es opcional y se utiliza para verificar errores en la transmisión de datos. Puede ser de paridad par o impar, donde la paridad par garantiza que el número total de bits 1 en el data frame sea par, y la paridad impar asegura que sea impar. Si hay una discrepancia, el UART receptor detecta un error en la transmisión. Para iniciar la transmisión de datos, el UART transmisor envía un bit de inicio (*Start Bit*) cambiando la línea de datos de alto a bajo. Al finalizar el envío de datos, se envían uno o dos bits de parada (*Stop Bits*) para volver a llevar la línea de datos a un estado alto. [12]

Figura 7.2: Empaquetado de bits protocolo UART. [12]

Según [12], este es el procedimiento de Transmisión de UART:

1. El UART transmisor recibe los datos en forma paralela desde el bus de datos del sistema.
2. El UART transmisor prepara el paquete de datos añadiendo un bit de inicio, el bit de paridad y los bits de parada a los datos recibidos.
3. El paquete de datos completo se envía de manera serial, comenzando con el bit de inicio, seguido por los bits de datos, el bit de paridad y finalizando con los bits de parada. Durante esta transmisión, el UART receptor muestrea la línea de datos al ritmo del *baud rate* preconfigurado para recibir los bits de manera adecuada.
4. El UART receptor descarta el bit de inicio, el bit de paridad y los bits de parada del marco de datos recibido, conservando únicamente los bits de datos originales.
5. El UART receptor convierte los datos recibidos de forma serial de nuevo a forma paralela y los transfiere al bus de datos de salida del sistema receptor.

Se utilizará una computadora con sistema operativo Linux. Para utilizar los puertos seriales se hace uso del programa *minicom*, para esto, primero se debe ejecutar en la terminal *dmesg* mientras algún dispositivo está conectado al USB, este comando dará una salida donde se indica un puerto, por ejemplo ttyUSB0. Ya obtenido el nombre del puerto a utilizar, se configura a minicom para que utilice este puerto y las demás configuraciones necesarias para la comunicación serial (baudios, paquete de datos, etc) y se guardan las configuraciones. Una vez configurado minicom se ejecuta en la terminal con el comando *sudo minicom*, esto abrirá una terminal mostrando la actividad en ese puerto. [13]

8. PREGUNTA 8

Investigue el funcionamiento básico del controlador ST7789V de la pantalla LCD RGB de la tang nano 9k. La hoja de datos será entregada por el profesor del curso.

El ST7789V se encarga de recibir datos, procesarlos y controlar directamente los píxeles de la pantalla LCD. [14]
Este controlador distingue entre comandos y datos. Los comandos configuran el controlador, mientras que los datos son los valores que se escriben en la memoria de la pantalla para definir los colores de los píxeles. Para diferenciar entre comandos y datos, se utiliza la señal DCX (*Data/Command*). Cuando esta señal está en bajo, el valor que se envía es un comando, cuando está en alto, el valor que se envía es un dato. [14]
Al inicio el controlador necesita ser configurado. Para esto, se envían comandos de inicialización a través del SPI. Después de la inicialización, la FPGA puede comenzar a enviar los datos de la imagen para llenar la memoria de la pantalla. Los datos de la imagen se envían secuencialmente, primero configurando las regiones de la pantalla donde se va a dibujar y luego enviando los valores de color. [14]
Una vez configurada la pantalla y cargados los datos de imagen, el controlador se encarga de manejar los píxeles de la pantalla LCD de acuerdo con los valores que se han enviado. Estos valores permanecen en la memoria interna del controlador hasta que se actualicen con nuevos datos. [14]

REFERENCIAS

[1] D. M. Harris y S. L. Harris, *Digital Design and Computer Architecture*, RISC-V Edition. San Francisco, CA: Morgan Kaufmann, 2022.

[2] J. Williams, Çlock Management Techniques in FPGA Designs,"*IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 65, no. 5, pp. 598-602, May 2018.

[3] Xilinx Inc., Çlock Enables and Their Applications in FPGA Designs,"*Xilinx Application Note*, AN-1245, 2020.

[4] A. Brown and P. Gupta, .Efficient Clock Gating and Frequency Division in FPGAs,ïn *Proceedings of the International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 321-326, Sep. 2019.

[5] AMD Customer Community, Xilinx.com, 2024.

[6] M. H. Rashid, "Debouncing Circuits for Digital Input Systems,"*IEEE Circuits and Systems Magazine*, vol. 13, no. 4, pp. 48-53, Nov. 2017.

[7] K. Y. Lee, "Metastability in Digital Systems: Causes, Effects, and Solutions,"*IEEE Journal of Solid-State Circuits*, vol. 50, no. 6, pp. 1473-1480, Jun. 2015.

[8] A. Smith, "Design and Implementation of Synchronizers for Asynchronous Inputs,"*IEEE Design & Test of Computers*, vol. 29, no. 3, pp. 72-79, Jun. 2018.

[9] "VHDL debouncer circuit,.[El]ectrical Engineering Stack Exchange, May 20, 2012.

[10] P. Dhaker, *Introduction to SPI Interface*, Analog Dialogue, Sep-2018.

[11] F. Leens, *An Introduction to I2C and SPI Protocols*, IEEE Instrumentation & Measurement Magazine, Feb-2009.

[12] E. Peña and M. G. Legaspi, *UART: A Hardware Communication Protocol Understanding Universal Asynchronous Receiver/Transmitter*, Analog Dialogue, Dec-2020.

[13] *How To Configure Minicom To Connect Over USB Serial UART*, [Online]. Available: https://bloggerbust.ca/post/how-to-configure-minicom-to-connect-over-usb-serial-uart.

[14] *ST7789V*, Sitronix Technology Corporation, 2013.