

UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

PROGRAMA DE FORMACIÓN DUAL

FACULTAD DE INGENIERÍA

REPORTE DE ACTIVIDADES

*PROYECTO DE SOFTWARE WINDOW LIFTER*

MENTORES:

Alejandro Rivera Garay

Continental Automotive - Querétaro Campus

Mariano Garduño

Universidad Autónoma de Querétaro

APRENDICES:

Fuentes Flores Lorena

Martínez Olvera Judith

Vite Gonzales Cynthia

Espinosa Bernal Giovanni

Ugalde Romero Dulce Carolina

Santiago de Querétaro, Qro., 12 de Diciembre de 2019.

ÍNDICE

[**I.** **INTRODUCCIÓN** 3](#_Toc24546506)

[**II.** **OBJETIVOS** 3](#_Toc24546507)

[**Objetivo general** 3](#_Toc24546508)

[**III.** **MARCO TEÓRICO** 3](#_Toc24546509)

[**IV.** **MATERIALES** Y **EQUIPO** 4](#_Toc24546510)

[**V.** **METODOLOGÍA** 5](#_Toc24546511)

[**VI.** **RESULTADOS** 5](#_Toc24546512)

[**VII.** **CONCLUSIONES** 7](#_Toc24546513)

1. **INTRODUCCIÓN**

Se busca simular el funcionamiento de una ventana automática de un automóvil. Se utilizarán dos tarjetas de desarrollo NXP y diversos componentes electrónicos para la ejecución de este proyecto. Este proyecto emplea dos botones del tipo “Push Button” (presentes en la tarjeta NXP) como simulación de los botones que se encuentran en los automóviles empleados para subir y bajar las ventanas; el efecto de estos botones, uno para subir el vidrio y el segundo para bajarlo, se podrá observar en un conjunto de 4 LEDs. Estos LEDs serán la representación de los actuadores (motores) empleados para realizar el movimiento de los vidrios dentro de un automóvil, mostrarán el movimiento simulado de una ventana. De igual manera, se tendrán indicadores (LEDs en la tarjeta NXP) para marcar el inicio y el fin de la carrera de la ventana, es decir, cuando la ventana este completamente abierta o completamente cerrada. Todo este sistema funcionará con el protocolo de comunicación SPI entre las dos tarjetas NXP.

1. **OBJETIVOS**

**Objetivo general**

Emplear la tarjeta de desarrollo NXP con comunicación serial SPI para simular por medio del uso de LEDs y Push Buttons el funcionamiento “Window Lifter” de un automóvil.

**Objetivos específicos**

* Definir los requerimientos necesarios para la realización del proyecto, con el fin de tener una base para el desarrollo organizado del mismo.
* Elaborar los diagramas necesarios para establecer gráficamente el funcionamiento del sistema.
* Emplear la comunicación SPI configurando las dos tarjetas de desarrollo NXP disponibles para ejecutar las acciones requeridas para el funcionamiento del proyecto.
* Diseñar y realizar el esquema electrónico y de software propuesto para el sistema.
* Programar en ambas tarjetas NXP el diseño de funcionamiento propuesto.
* Verificar el correcto funcionamiento del sistema de acuerdo a lo desarrollado en los requerimientos.

1. **MARCO TEÓRICO**

**Comunicación serial**

Este concepto se refiere a la transmisión bit por bit de cada byte y a pesar de su baja velocidad de transmisión comparada con la comunicación paralela, es más usada porque es más barata (sólo un cable), y llega a tener velocidades de transmisión considerables. Dentro de la comunicación serial existen distintos tipos por su aplicación, ejemplos de ellos son I2C, SPI, UART, cada uno con su respectivo protocolo.

**Comunicación serial**

Del acrónimo “Serial Peripheral Interface”

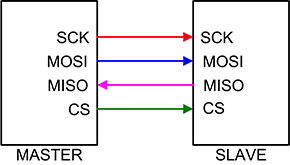
Principales características:

|  |  |
| --- | --- |
| Cables usados | 4 |
| Velocidad Máxima | Más de 10 Mbps |
| Sincronía | Síncrono |
| ¿Serial o Paralelo? | Serial |
| # máximo de maestros | 1 |
| # máximo de esclavos | No limitado en teoría |

Generalmente este protocolo de comunicación tiene sus aplicaciones en distancias pequeñas y por su rapidez, para guardar datos en memoria

**Configuración de cables**

Los cuatro cables usados en la comunicación serial entre tarjeta master y slave [fig 3.3.1] consiguen una transmisión de datos modo full dúplex, es decir, el maestro recibe y envía datos casi al mismo tiempo.



***Figura 1.*** *Cables empleados en el protocolo SPI entre maestro y esclavo.*

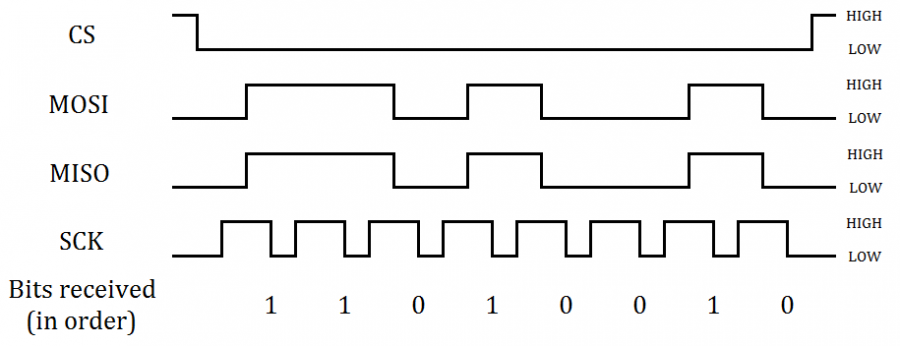
Las señales de datos, MISO y MOSI, corresponden a Master In, Slave Out y Master Out, Slave In respectivamente; SCK corresponde al reloj de sincronización y CS al Chip Select o seleccionador de esclavo ya que puede haber más esclavos.

Cabe reslatar que el Master controla las señales de control, por tnto controla la totalidad de la comunicación.

El cable CS es puesto en alto cuando el maestro desea comunicarse con el esclavo conectado a este mismo, puede haber número ilimitado de esclavos en teoría.

**Envío de datos**

El proceso con base en la figura 3.4.1. Comienza cuando el Maestro de nuestra comunicación baja la línea CS seleccionando así al esclavo correspondiente, después el maestro manda una señal de reloj consistente. Cuando el esclavo detecta un bajo en su línea de CS, comienza a enviar (Línea MISO) o recibir (Líne MOSI) datos



***Figura 2***. Datos en el bus SPI.

**Arquitectura de software**

La arquitectura de software depende del hardware, tanto más complejo sea este, más capas requiere la arquitectura para darle mayor soporte al sistema e interfaces más adecuadas para el usuario. (1)

A continuación se muestran las capas que componen la arquitectura de un sistema de software embebido.

**Capa de aplicaciones**

Ubicada en el nivel superior de la jerarquía de software, implementa la funcionalidad del sistema, tiene como objetivo realizar funciones del sistema.

**Servicios**

Interfaz del servicio que el sistema operativo proporciona a la aplicación

**Sistema operativo**

Sistema de software que permite administrar uniformemente los recursos del hardware, resume sus funciones y las proporciona a las aplicaciones en forma de servicios.

**Capa de abstracción**

Forma parte del software, entre el hardware del sistema integrado y el sistema operativo. Incluye el gestor de arranque, el paquete de soporte de la placa “BSP” los controladores del dispositivo y otros componentes.

**Drivers**

Programa informático que permite al sistema operativo interactuar con periféricos.

**Ciber seguridad**

Se refiere a la protección del sistema de elementos externos a través del seguimiento de normas sobre programación.

**Desbordamiento de variables**

Consiste en la planificación sobre la escritura en variables para evitar el sobredimensionamiento de las mismas, así como las respectivas protecciones ante esta posibilidad.

1. **MATERIALES Y EQUIPO**
2. Tarjeta de desarrollo NXP tipo: S32K144EVB-Q100
3. Analizador de señales SALEAE
4. Cables duponts
5. Software S32 Design Studio for ARM Version 2018.R1
6. **METODOLOGÍA**

1. Definición del proyecto de software

2. Análisis de requerimientos de los entregables

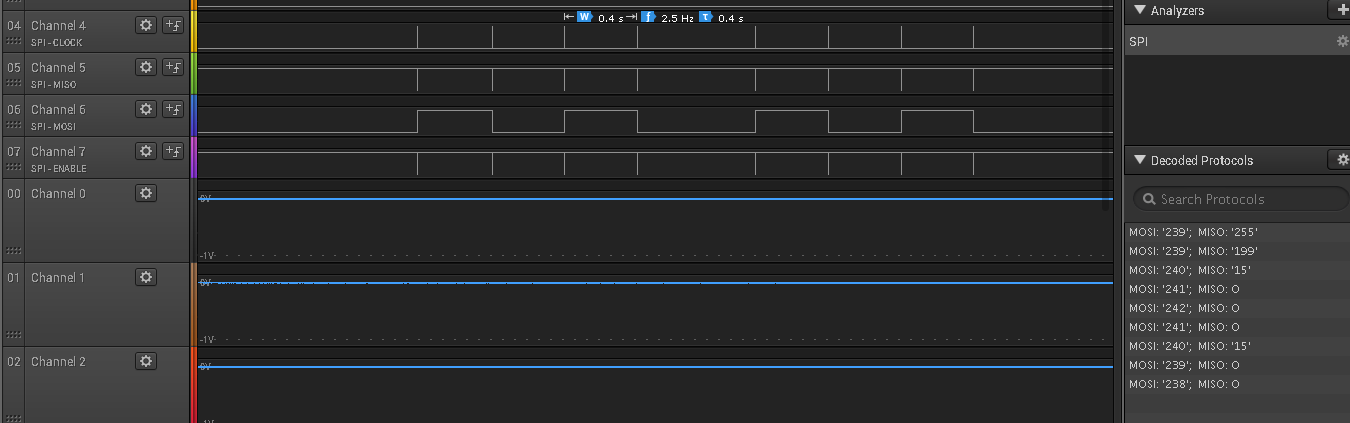
3. Diseño de la máquina de estados

4. Diseño de la arquitectura del software

5. Construcción del código

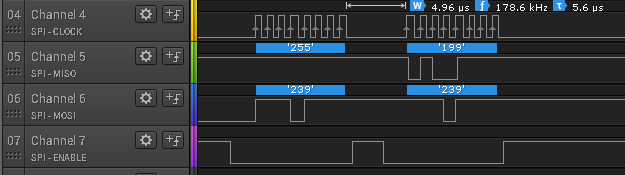
6. Validación con pruebas de casos

1. **RESULTADOS**



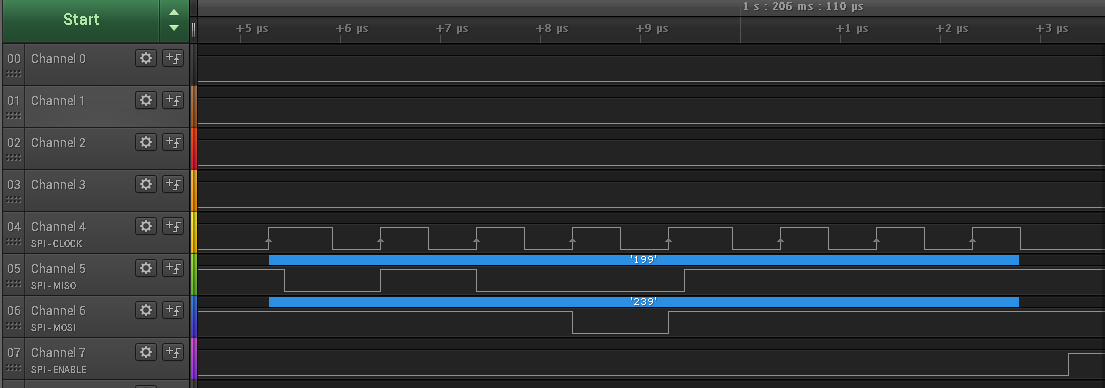
***Figura 3***. Análisis de datos en el bus SPI.

Figra 2: Señales analizadas tanto en transmisor como en receptor (MOSI, MISO) en esquina inferior izquierda



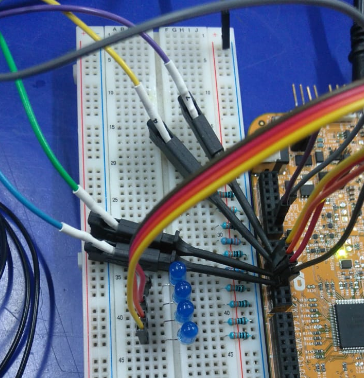
***Figura 4***. Analisis del chip select (CS).

Figura 4: Número de datos enviados por cada Frame: 1



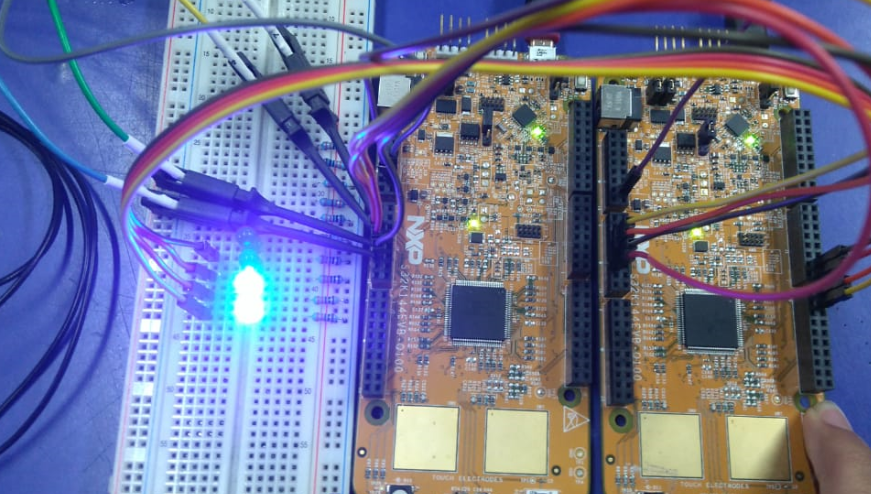
***Figura 5***. Análisis de la señal de reloj (CLK).

Figura 5: La señal del reloj comienza en bajo y sube por lo que la polaridad es 0, además, la captura del dato ocurre medio ciclo después del comienzo del reloj por lo que es una fase de 1.



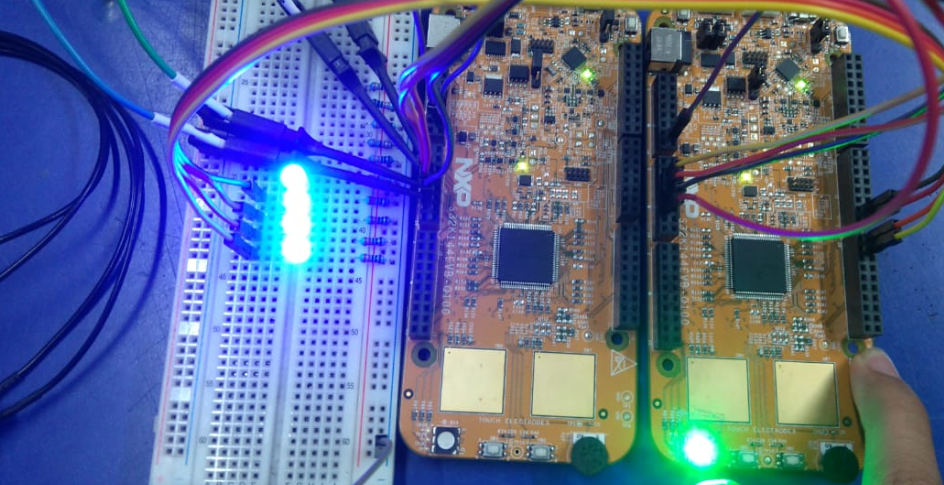
***Figura 6***. Conexión física de los duponts en la tarjeta master.

Figura 6: Conexión correcta de los duponts morado, amarillo, verde y azul en los pines del puerto b 17,14, 15 y 16 que corresponden a la interfaz del analizador lógico.



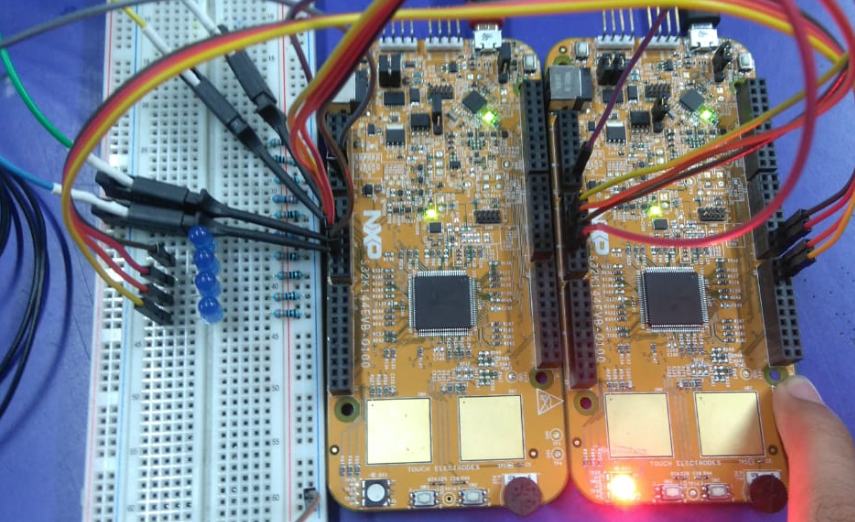
***Figura 7***. Estado intermedio de los leds.

Figura 7: Los leds se encuentran en un estado intermedio, no arriba ni abajo.



***Figura 8***. Estado ventana arriba totalmente

Figura 8: Todos los LEDs se encuentran encendidos y el LED de la tarjeta slave está en color verde por lo que la ventana está completamente arriba.



***Figura 9***. Estado ventana abajo totalmente

Figura 9: Todos los LEDs se encuentran apagados y el LED de la tarjeta slave está en color rojo por lo que la ventana está completamente abajo.

1. **REQUERIMIENTOS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REQUERIMIENTOS FUNCIONALES** | | | | |
| Req. | Descripción del requerimiento | Test Case | Estado | Resultado |
|  | ***Componentes (localización y especificaciones)*** |  |  |  |
|  | Establecer comunicación mediante protocolo de SPI. | Verificación de la configuración del código principal. |  |  |
|  | Se tendrá un **Push Button 1 (PB1)** para representar la acción *WindowUp* y un **Push Button 2 (PB2)** para representar la acción *WindowDown.* | Deberán ser visibles y serán representados por los botones SW2 (para el **PB1**) y SW3 (para el **PB2**) en la tarjeta NXP configurada como *master* . | Passed |  |
|  | Se tendrá un **Conjunto de LEDs (CLED1)** el cual consiste en cuatro LEDs externos a la tarjeta colocados en fila. Estos LEDs simularán el movimiento automático del vidrio de la ventana de un automóvil.  Serán numerados dependiendo de su posición en el circuito siguiendo el orden de izquierda a derecha: D1, D2, D3 y D4. | Se localizarán de manera externa a la tarjeta. Deberán ser visibles, de color azul, de tipo ultra brillantes de 5mm. | Passed |  |
|  | Se utilizará el LED marcado como D11 de la tarjeta NXP **(D5)**. Este LED se utilizará de tres modos distintos:  **D5** encenderá de color ROJO *(D5\_R) -*utilizando el LED de la tarjeta DXT marcado como TP30- dependiendo de la condición señalada en el Req.8\*  **D5** encenderá de color VERDE *(D5\_V)* *-*utilizando el LED de la tarjeta DXT marcado como TP29- dependiendo de la condición señalada en el Req.9\*  **D5** encenderá de color AZUL *(D5\_A)* *-*utilizando el LED de la tarjeta DXT marcado como TP28- dependiendo de la condición señalada en el Req.13\*  \*Ver Req.8, Req.9 y Req.13 | Sera visible en la tarjeta NXP.  Se observará que el color corresponda con lo establecido en este requerimiento. | Passed |  |
|  | ***Funcionamiento de Componentes*** |  |  |  |
|  | Al presionar **PB1** por más de 50mS con una tolerancia de ±10mS, y al mantenerlo presionado **CLED1** mostrará la activación secuencial de los 4 LEDs de la siguiente manera: D1, D2, D3 y por ultimo D4 (orden ascendente). | Se verificará su funcionamiento al observar el encendido secuencial de los 4 LEDs que integran **CLED1.** | Passed |  |
|  | La secuencia de encendido de **CLED1** (ver Ref. 4)se efectuará mientras **PB1** se mantenga presionado. Y se detendrá hasta completar la cantidad total de LEDs si **PB1** continúa presionado. Esto para simular que el vidrio está subiendo (*WindowUp*).  En caso de que se dejara de presionar **PB1** (si no se encendieron los 4 LEDs), la secuencia de encendido de **CLED1** se detendrá y se mantendrán activos los LEDs encendidos con anterioridad. Esto para simular que el vidrio no subió completamente. | Se verificará su funcionamiento al observar el encendido secuencial de los 4 LEDs de **CLED1**, dependiendo del uso de **PB1.** | Passed |  |
|  | Al presionar **PB2** por más de 500mS con una tolerancia de ±200mS, y al mantenerlo presionado **CLED1** mostrará la desactivación secuencial de los 4 LEDs de la siguiente manera: D4, D3, D2 y por ultimo D1 (orden descendente). | Se verificará su funcionamiento al observar el apagado secuencial de los 4 LEDs que integran **CLED1.** | Passed |  |
|  | La secuencia de apagado de **CLED1** (ver Ref. 4)se efectuará mientras **PB2** sea presionado. Y detendrá hasta completar la cantidad total de LEDs si **PB2** continúa presionado. Esto para simular que el vidrio está bajando (*WindowDown*).  En caso de que se dejara de presionar **PB2** (si no se apagaron los 4 LEDs), la secuencia de apagado de **CLED1** se detendrá y se mantendrán activos los LEDs encendidos con anterioridad. Esto para simular que el vidrio no bajó completamente. | Se verificará su funcionamiento al observar el apagado secuencial de los 4 LEDs de **CLED1**, dependiendo del uso de **PB2**. | Passed |  |
|  | El LED **D5** (ver Req.3)encenderá de color VERDE al momento en el que la ventana este completamente arriba y ya no sea posible seguir subiendo.  Esto se basará de acuerdo al Req.5, es decir: cuando la acción *WindowUp* concluya y todos los LEDs de **CLED1** estén encendidos, **D5** se activará de color verde: *D5\_V.* | Se observará que el color de **D5** sea verde cuando la acción *WindowUp* concluya. | Passed |  |
|  | El LED **D5** (ver Req.3)encenderá de color ROJO al momento en el que la ventana este completamente abajo y ya no sea posible seguir bajando.  Esto se basará de acuerdo al Req.5, es decir: cuando la acción *WindowDown* concluya y todos los LEDs de **CLED1** estén apagados, **D5** se activará de color rojo: *D5\_R.* | Se observará que el color de **D5** sea rojo cuando la acción *WindowDown* concluya. | Passed |  |
|  | En caso de que los dos botones, **PB1** y **PB2** se presionen al mismo tiempo en cualquier momento de la ejecución por más de 600mS con una tolerancia de ±200mS, **D5** encenderá de color azul *D5\_A*, señalando un error en la ejecución. Y el sistema entrará en estado *idle*, es decir, mantendrá el estado anterior a este evento. *Error.* | Se observará que el sistema se mantendrá en el último estado de la ejecución, asimismo se observara que el **D5** encienda de color Azul. | failed daba muchos problemas a las interrupciones externas, por la logica de programacion |  |
|  | En caso de que ninguna de las condiciones establecidas en los Req.8 y Req.9 se cumpla, el LED **D5** permanecerá desactivado. | Se observará el funcionamiento de este LED durante la ejecución. Y no deberá de encender a menos de que los criterios anteriores se cumplan. | Passed |  |

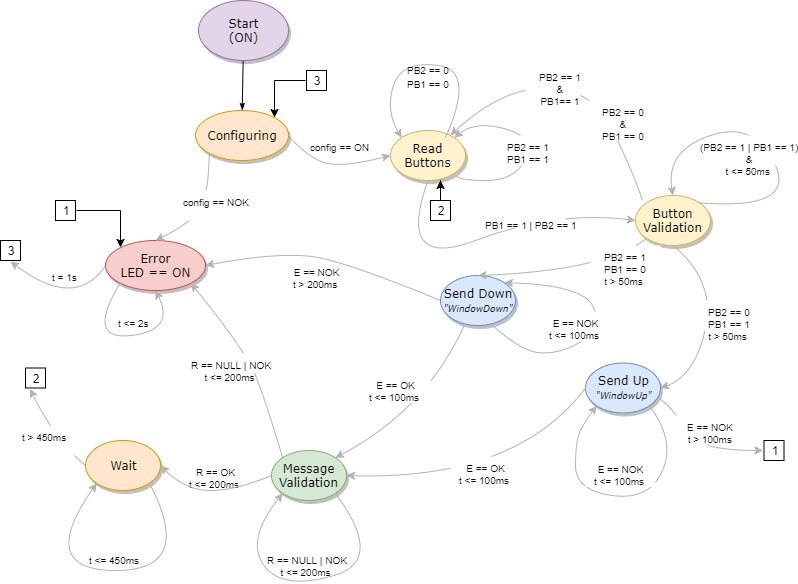
Tabla 1. Desglose de requerimientos funcionales.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **REQUERIMIENTOS NO FUNCIONALES** | | | | |
| Req. | Descripción del requerimiento | Test Case | Estado | Resultado |
|  | ***Comunicación*** |  |  |  |
|  | Velocidad del reloj de comunicación (pin SCK) a 10 MHz con tolerancia ±5%. | Revisar línea de SCL (pin 67) con el analizador lógico de señales. Verificar que la velocidad de transmisión esté en el rango del requerimiento. | Follow up |  |
|  | Número de bits: 8 | Revisar que se cumplan la cantidad de bits en la línea de MOSI y MISO (pin 64 y 65) con el analizador lógico de señales. | Passed |  |
|  | Paridad: Ninguna | Revisar línea de MOSI y MISO (pin 64 y 65) con el analizador lógico de señales. Y verificar que no se tenga bit de paridad en la señal. | Passed |  |
|  | Fase: 1 | Revisar línea de MOSI y MISO (pin 66 y pin 65) con el analizador lógico de señales y verificar que el muestreo del dato se realice cuando el reloj cambia de estado de alto a bajo. | Passed |  |
|  | Polaridad: 1 | Revisar línea de SCL(pin 67 ) con el analizador lógico de señales y verificar que la línea de SCL en estado esté en bajo. | , passed se cumple pero el requisito está mal planteado |  |
|  | ***Conexiones*** |  |  |  |
|  | La comunicación serial SPI será de la Tarjeta 1 (*Master*) a la Tarjeta 2 (*Slave*). | Verificación de la configuración de ambas tarjetas en el código. | Passed |  |
|  | El **PB1** y el **PB2** serán los Push Buttons integrados en la tarjeta NXP 1 (*Master*): SW3 y SW2.  El LED **D5** será el ya integrado a la tarjeta NXP 1 (*Master*): D11. | Se verificará que ambos botones y el LED **D5** sean funcionales de acuerdo con los requerimientos establecidos. | follow up, el requisito está mal escrito |  |
|  | El conjunto de LEDs **CLED1** se conectarán a las salidas de la Tarjeta NXP 2 (*Slave*).  Estos LEDs serán externos a la tarjeta y se conectarán empleando las resistencias necesarias para adecuar a ellos el voltaje que entrega la Tarjeta NXP en sus salidas físicas. | Se verificará que todos los LEDs externos que componen **CLED1** sean funcionales de acuerdo a lo establecido. | Passed |  |
|  | ***Características eléctricas*** |  |  |  |
|  | Se utilizarán LEDs color azul de 5mm de diámetro, voltaje de 3.2  a 3.4V y 20mA conectados con una resistencia de protección de 82Ω para un voltaje máximo de 5V (+- 200mV) a salida de la tarjeta o sin resistencia para para un voltaje mínimo de 3.3V (+- 100mV) a salida de la tarjeta. | Verificar la conexión del led de acuerdo al voltaje de salida de la tarjeta. | follow up, algunas cosas se cumplen y otras no |  |
|  | El botón **PB1** deberá ser implementado con un circuito pull down. | Verificar el circuito de pull down para el botón **PB1** en la descripción de la tarjeta. | follow up, revisar datasheet |  |
|  | El botón **PB2** deberá ser implementado con un circuito pull down. | Verificar el circuito de pull down para el botón **PB2** en la descripción de la tarjeta. | follow up, revisar datasheet |  |
|  | El voltaje máximo de entrada del sistema serán 5V y el mínimo 3.3V (+- 200mV) de acuerdo a lo requerido. | Verificar la alimentación de entrada de los pines configurados como entradas. | Passed |  |
|  | El voltaje y la corriente de alimentación de entrada para el sistema deberá ser suministrados por un puerto USB de computadora(voltaje máximo de entrada del sistema son 5V y 1A) | Revisar que las tarjetas sean conectadas a algún puerto USB | Passed |  |

Tabla 2, Desglose de requerimientos no funcionales.

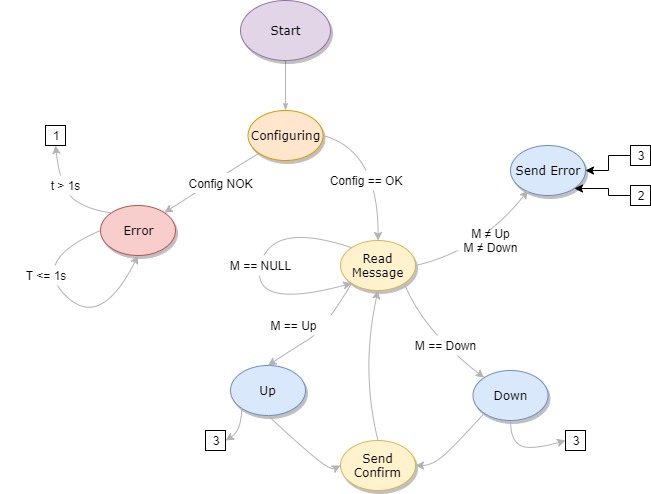
1. **MÁQUINA DE ESTADOS**

En la figura 7 se describe la Máquina de estados de la tarjeta master del proyecto.



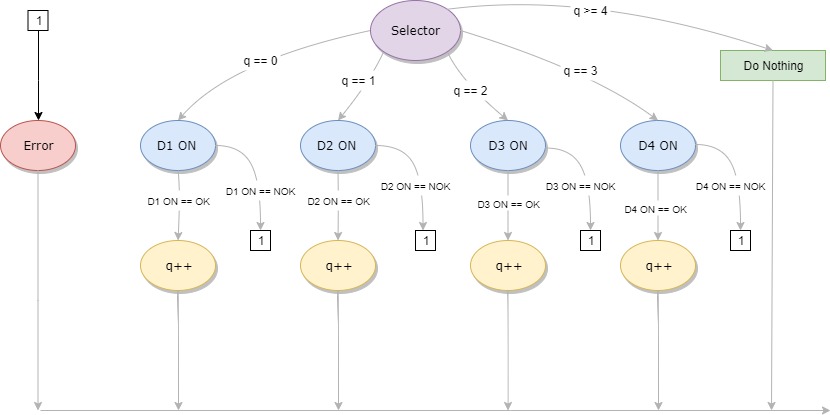
**Figura 7**. Máquina de estados en la tarjeta master.

En la figura 8 se describe la Máquina de estados de la tarjeta master del proyecto.



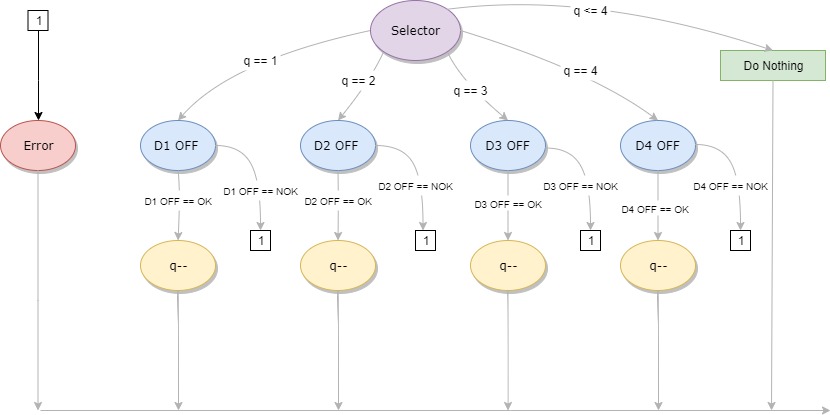
***Figura 8***. Máquina de estados de la tarjeta slave

En la figura 9 se describe la Máquina de estados para el movimiento ascendente de los LEDs.



***Figura 9***. Máquina de del movimiento ascendente de los LEDs

En la figura 10 se describe la Máquina de estados para el movimiento descendente de los LEDs.



***Figura 10***. Máquina de del movimiento descendente de los LEDs

1. **CONCLUSIONES**

Se realizó una comunicación full dúplex en ambas tarjetas por lo que se cuidó que primero se transmitiera y luego se recibiera con código, ello provocó que no funcionara ninguna de las dos si todos los cables no estaban correctamente conectados. También el ordenar y encontrar los registros no fue muy sencillo al haber muchos de ellos y usar el código para configurarlos de manera correcta, también cuidamos los errores de sobre flujo al asegurarnos que nos transmitieran datos más grandes.