Práctica 2- Control puerta de garaje

Tabla de contenido

Práctica 2- Control puerta de garaje

Esta práctica tiene como objetivo simular el control automático de una puerta de garaje utilizando una placa con el módulo TM1638, que incluye 8 LEDs, botones y displays de 7 segmentos. La puerta se puede controlar de forma manual o automática mediante la interacción con los botones y las señales de los sensores, como el mando a distancia o las fotocélulas.

Objetivos:

1. **Controlar la apertura y cierre de la puerta**: La puerta puede estar completamente abierta, completamente cerrada, o en movimiento (abriéndose o cerrándose). El sistema debe gestionar estas transiciones de forma secuencial y visualmente representarlas mediante los LEDs y los displays.
2. **Simulación del tiempo de apertura y cierre**: Cada acción (apertura o cierre) tarda un tiempo determinado, que debe mostrarse en los displays. La práctica requiere la simulación de estos tiempos y su representación visual en la pantalla.
3. **Interacción mediante botones**:
    - **Botón azul B0 (Mando a distancia)**: Simula el control remoto de la puerta. Si la puerta está cerrada, pulsar este botón la abre, y viceversa.
    - **Botón B1 (Fotocélula)**: Este botón simula una señal de seguridad, similar a una fotocélula en la puerta de un garaje. Si la fotocélula detecta un obstáculo mientras la puerta se está cerrando, el movimiento de la puerta debe detenerse de inmediato.
4. **Control visual con LEDs y display**:
    - **LEDs**: Los LEDs simulan el movimiento de la puerta. Los primeros LEDs encendidos representan la puerta cerrada y los últimos LEDs encendidos representan la puerta abierta. A medida que la puerta se mueve, los LEDs cambian progresivamente de estado, indicando el movimiento de apertura o cierre.
    - **Display de 7 segmentos**: El display mostrará una cuenta regresiva que simula el tiempo restante para completar la operación de apertura o cierre. Cuando la puerta está completamente abierta o cerrada, el display mostrará un valor indicativo de que la acción ha finalizado.
5. **Simulación de las condiciones de la puerta**:
    - **Puerta cerrada**: Todos los LEDs están apagados, indicando que la puerta está completamente cerrada.
    - **Puerta abriéndose**: Los LEDs se encienden progresivamente, simulando la apertura de la puerta.

- **Puerta abierta**: Todos los LEDs están encendidos, indicando que la puerta está completamente abierta.
- **Puerta cerrándose**: Los LEDs se apagan progresivamente, simulando el cierre de la puerta.

6. **Parada automática**: El sistema debe detener automáticamente la puerta si se detecta un obstáculo (simulado por el botón B1) mientras la puerta se está cerrando.

Descripción de la funcionalidad:

1. Para abrir o cerrar la puerta se utiliza un mando a distancia (B0-Botón azul en la placa). Un pulso positivo (transición de estado 0 a 1) indica que se quiere iniciar un movimiento (cierra o abre). El sentido del movimiento siempre será el contrario al último que se haya efectuado. Si anteriormente estaba cerrando, se abre, y viceversa.

2. Cuando está en movimiento, la puerta se para por las siguientes condiciones:
   - Un pulso positivo en el mando a distancia.
   - Fin de recorrido. Si está abriendo, la puerta está completamente abierta. Si está cerrando, la puerta está totalmente cerrada.
   - Célula fotoeléctrica. Si durante la secuencia de cierre se detecta algún obstáculo por el detector de presencia (B1), la puerta debe pararse inmediatamente y solo debe moverse cuando no haya obstáculo.

3. Para mover la puerta, el motor se comanda por 2 salidas: ADELANTE (Led 8), RETROCESO (Led 1). Si las 2 están a 0, el motor se para y la puerta también en consecuencia.

4. Para detectar el fin de recorrido, se simulará la posición de la puerta suponiendo que esta tiene que recorrer una distancia máxima de 3m en 10 segundos. (3cm cada décima de segundo), tanto para abrir como para cerrar.
   La posición de cierre total es 3m y la apertura total es 0m. La posición debe calcularse continuamente en función del movimiento.

5. El tiempo en segundos de apertura y cierre debe ser visible en el display de 7 segmentos, con una cuenta regresiva durante el movimiento.

Tareas:

1. Detección de flanco. Estado actual=1 y estado anterior=0
2. Parada y cambio de dirección por flanco. Activar en dirección contraria o parar motor.
3. Cálculo de movimiento. Usar retardo de 1 décima de segundo. (100ms).
   - Visualizar tiempo restante en los displays.
4. Parada por final de curso.
5. Parada por fotocélula.

Librerías

La biblioteca **TM1638** en los archivos TM1638.c y TM1638.h está diseñada para interactuar con el módulo de pantalla y teclas TM1638, proporcionando funciones clave para manejar los pines, leer botones, controlar los LEDs, y gestionar los displays de 7 segmentos. Esta biblioteca completa permite gestionar todas las interacciones con el módulo TM1638, proporcionando control sobre los LEDs, el display de 7 segmentos, y la lectura de los botones.

TM1638.h

Este archivo contiene las definiciones y las declaraciones de las funciones para manejar el módulo TM1638. A continuación se listan las funciones principales:

1. **void tm1638_TurnOn**(TM1638 *tm, uint8_t brightness)
    - **Descripción**: Enciende el display con el brillo especificado.
    - **Parámetros**:
        - tm: Puntero a la estructura TM1638.
        - brightness: Nivel de brillo (entre 0 y 7).

2. ***void tm1638_DisplayClear****(TM1638 *tm)*
    - **Descripción**: Limpia el display de 7 segmentos.
    - **Parámetros**: Puntero a la estructura TM1638.

3. ***void tm1638_Led****(TM1638 *tm,* ***int*** *position,* ***int*** *on)*
    - **Descripción**: Controla el estado de un LED en una posición específica.
    - **Parámetros**:
        - tm: Puntero a la estructura TM1638.
        - position: Número del LED (de 1 a 8).
        - on: 1 para encender, 0 para apagar.
4. ***void tm1638_DisplayTxt****(TM1638 *tm,* ***char*** **c)*
    - **Descripción**: Muestra una cadena de texto en el display de 7 segmentos.
    - **Parámetros**:
        - tm: Puntero a la estructura TM1638.
        - c: Cadena de texto a mostrar.

5. ***void tm1638_DisplayChar****(TM1638 *tm,* ***int*** *position,* ***char*** *c, bool dot)*
    - **Descripción**: Muestra un solo carácter en una posición del display, con la opción de activar el punto decimal.
    - **Parámetros**:
        - tm: Puntero a la estructura TM1638.
        - position: Posición del carácter (de 1 a 8).
        - c: Carácter a mostrar.
        - dot: 1 para activar el punto decimal, 0 para desactivarlo.

6. *uint8_t* ***tm1638_ScanButtons****(TM1638 *tm)*
    - **Descripción**: Escanea y lee el estado de los botones del TM1638.
    - **Parámetros**: Puntero a la estructura TM1638.
    - **Devuelve**: Un byte que representa el estado de los botones.
7. *uint8_t* ***tm1638_ReadKey****(TM1638 *tm, uint8_t buttons)*
    - **Descripción**: Devuelve el número del botón que ha sido presionado.
    - **Parámetros**:
        - tm: Puntero a la estructura TM1638.
        - buttons: Estado de los botones leídos previamente con ScanButtons
    - **Devuelve**: El número del botón presionado.
8. bool **tm1638_KeyState**(TM1638 *tm, uint8_t buttons, **int** position)
    - **Descripción**: Devuelve el número del botón que ha sido presionado.
    - **Parámetros**:
        - tm: Puntero a la estructura TM1638.
        - buttons: Estado de los botones leídos previamente con ScanButtons
        - position: Posicion (1,8) del botón
    - **Devuelve**: Estado del botón leído (0,1)

TM1638.c

Este archivo contiene la implementación de las funciones declaradas en TM1638.h. Aquí es donde se definen las operaciones detalladas de control del TM1638, incluyendo la manipulación de los pines de GPIO, la transmisión de datos al display y la lectura de los botones.

Configuración de la Librería TM1638:

Para utilizar la librería TM1638 y controlar los LEDs, botones, y displays de 7 segmentos, primero es necesario definir la estructura TM1638 y asignar los parámetros correspondientes en el código. A continuación se muestra cómo hacerlo:

1. **Definir la Estructura TM1638**
    - Dentro del archivo de código principal, en la sección de **definición de variables** (entre los comentarios /* Private user code ----------*/ y /* USER CODE BEGIN 0 */), se debe definir la variable TM1638 TM de la siguiente manera:

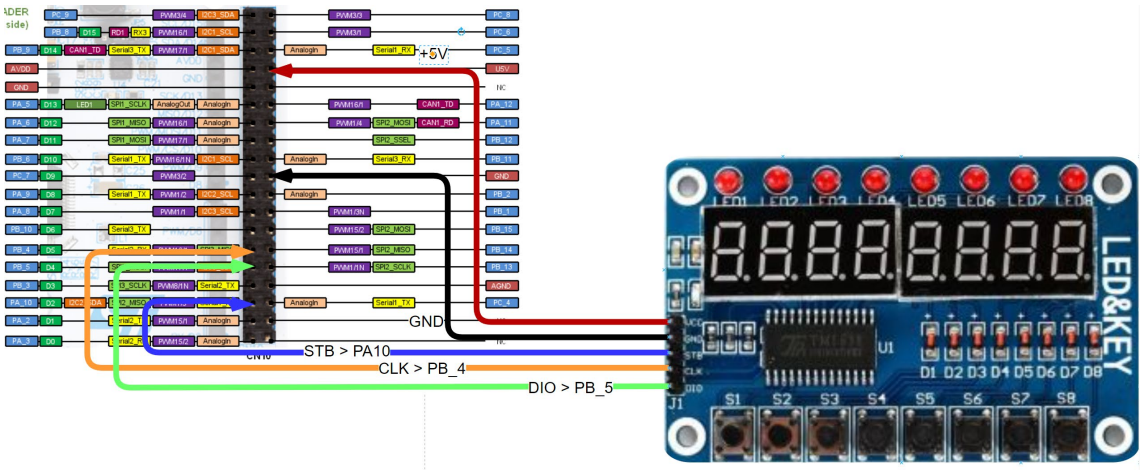
```
/* USER CODE BEGIN 0 */
TM1638 TM;
/* USER CODE END 0 */
```

2. **Asignar los Parámetros del TM1638**
    - Dentro de la función int main(void) del programa, se deben asignar los pines y puertos correspondientes al módulo TM1638 de la siguiente manera:

```
/* USER CODE BEGIN 2 */
TM.clk_port = CLK_GPIO_Port;
TM.clk_pin = CLK_Pin;
TM.stb_port = STB_GPIO_Port;
TM.stb_pin = STB_Pin;
TM.dio_port = DIO_GPIO_Port;
TM.dio_pin = DIO_Pin;
tm1638_TurnOn(&TM, 7); // Brillo del display = 7 (máximo)
tm1638_DisplayClear(&TM); // Limpiar el display
/* USER CODE END 2 */
```

- **tm1638_TurnOn(&TM, 7)**: Enciende el display y establece el nivel de brillo. En este caso, se utiliza el valor **7** para el brillo máximo.
- **tm1638_DisplayClear(&TM)**: Limpia el display, dejando todos los segmentos apagados.

Conexionado físico:

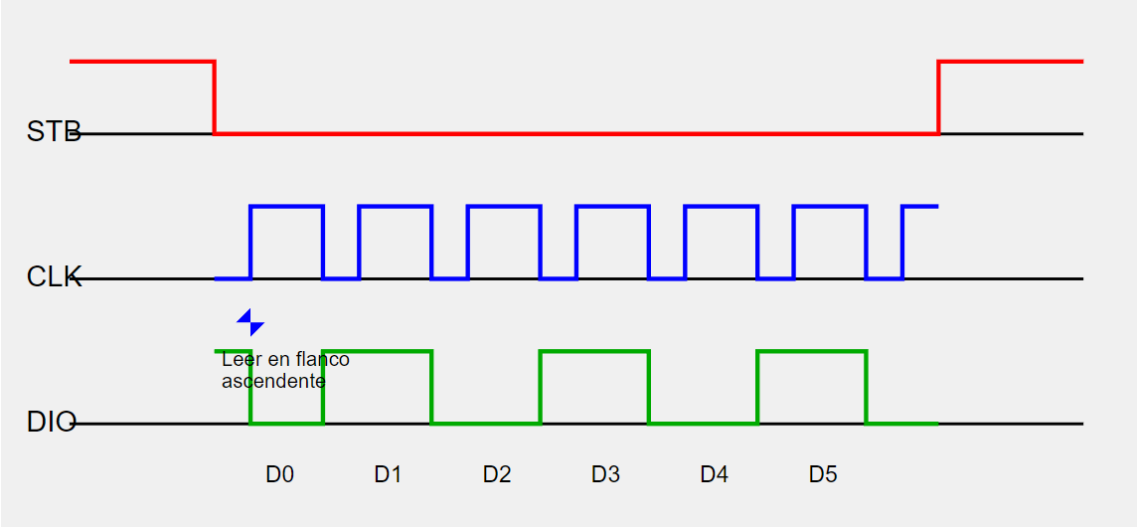


Protocolo de comunicación

El protocolo de comunicación del TM1638 funciona de la siguiente manera:

1. Inicio de la comunicación:
    - El maestro (por ejemplo, un microcontrolador) baja la línea STB (Strobe) a nivel bajo.
    - Esto indica el inicio de una nueva transmisión de comando o datos.
2. Transmisión de datos:
    - Los datos se envían a través de la línea DIO (Data Input/Output).
    - La transmisión ocurre en grupos de 8 bits (1 byte).
    - Los datos se envían en orden del bit menos significativo (LSB) al más significativo (MSB).
3. Sincronización:
    - Cada bit se transmite en sincronización con la señal de reloj (CLK).
    - El dato en DIO se lee en el flanco ascendente de CLK.
4. Fin de la comunicación:
    - Después de enviar todos los datos necesarios, el maestro vuelve a poner STB en alto.
    - Esto indica el final de la transmisión y hace que el TM1638 procese los datos recibidos.

5. Comandos y datos:
    - El primer byte enviado después de bajar STB generalmente es un comando.
    - Los bytes subsiguientes son datos, dependiendo del comando.



1. La línea roja representa STB (Strobe). Baja para iniciar la comunicación y sube para finalizarla.
2. La línea azul es CLK (Clock). Cada pulso de reloj corresponde a un bit de datos.
3. La línea verde es DIO (Data Input/Output). Muestra los niveles de los bits transmitidos.
4. Los bits se etiquetan como D0, D1, etc., transmitidos del menos significativo al más significativo.
5. Las flechas azules indican que los datos se leen en el flanco ascendente del reloj.

Este diagrama muestra la transmisión de 6 bits como ejemplo, pero en la práctica, el TM1638 opera con bytes completos (8 bits).

1. Distinción entre escritura y lectura: La principal diferencia entre la escritura y la lectura en el TM1638 se basa en el comando inicial y en la dirección del flujo de datos en la línea DIO.

    a) Escritura:
    - Se inicia con un comando de escritura.
    - El microcontrolador (maestro) envía datos al TM1638.
    - La línea DIO es controlada por el maestro durante toda la operación.

b) Lectura:
- Se inicia con un comando de lectura.

- Después del comando, el TM1638 toma el control de la línea DIO.
- El microcontrolador cambia la configuración de su pin DIO a entrada.
- El TM1638 envía los datos solicitados al microcontrolador.

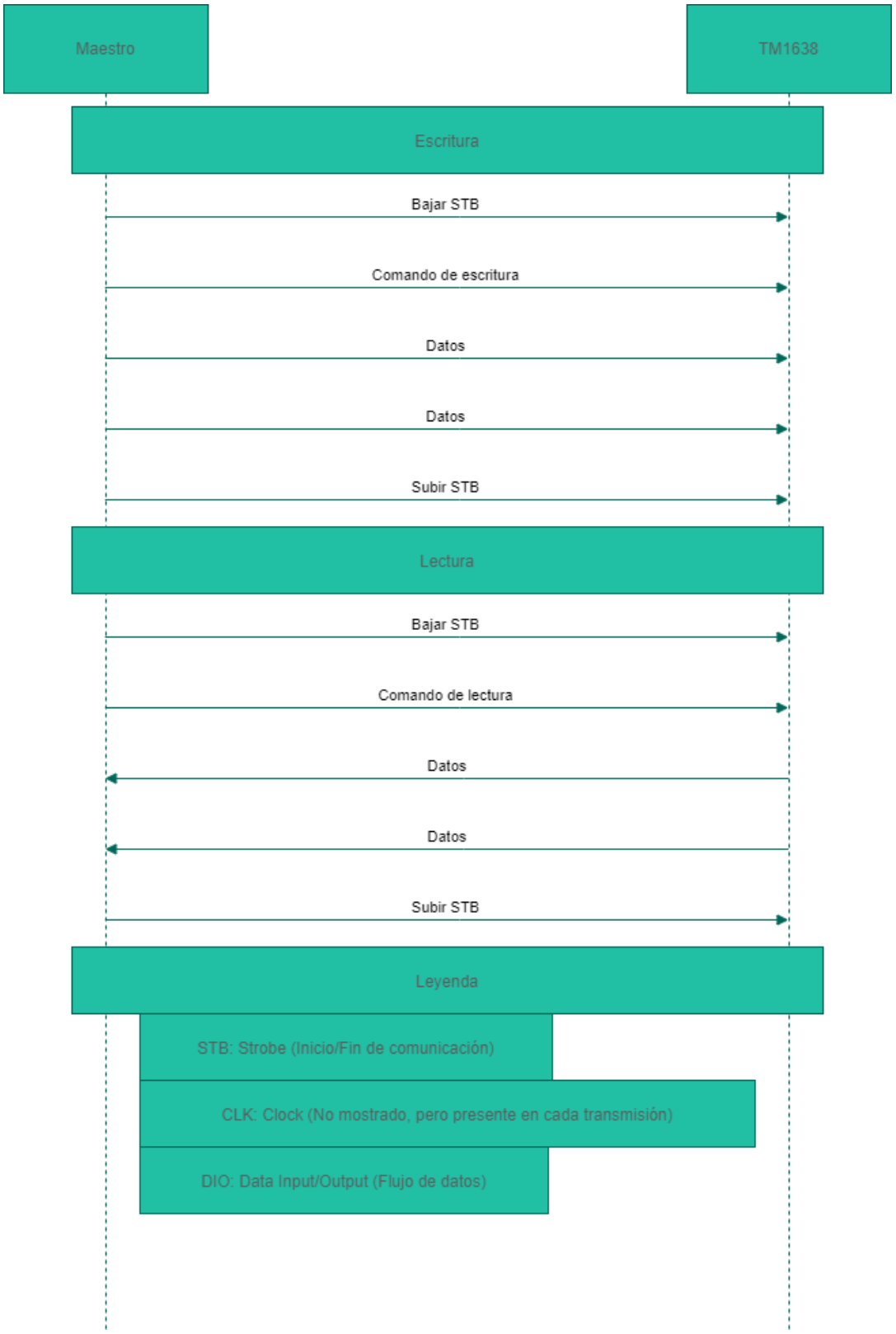
2. Proceso detallado:

Escritura:

1. El maestro baja STB.
2. El maestro envía el comando de escritura.
3. El maestro envía los datos.
4. El maestro sube STB para finalizar.

Lectura:

5. El maestro baja STB.
6. El maestro envía el comando de lectura.
7. El maestro cambia su pin DIO a modo de entrada.
8. El TM1638 envía los datos solicitados.
9. El maestro sube STB para finalizar.


Maestro
TM1638
Escritura
Bajar STB
Comando de escritura
Datos
Datos
Subir STB
Lectura
Bajar STB
Comando de lectura
Datos
Datos
Subir STB
Leyenda
STB: Strobe (Inicio/Fin de comunicación)
CLK: Clock (No mostrado, pero presente en cada transmisión)
DIO: Data Input/Output (Flujo de datos)