Alumno: Daniel Marquez

Plataforma embebida: EDU-CIAA-NXP
Link Github: https://github.com/damf618/Dispositivo-Primario

Aplicación:

Monitoreo del estado de las teclas TEC1, TEC2 y TEC3, teclas de alarma, falla y normal respectivamente, el usuario será notificado a través de los leds de la plataforma según el siguiente código de colores:

- Verde → Normal.
- Amarillo → Falla .
- Rojo → Alarma.

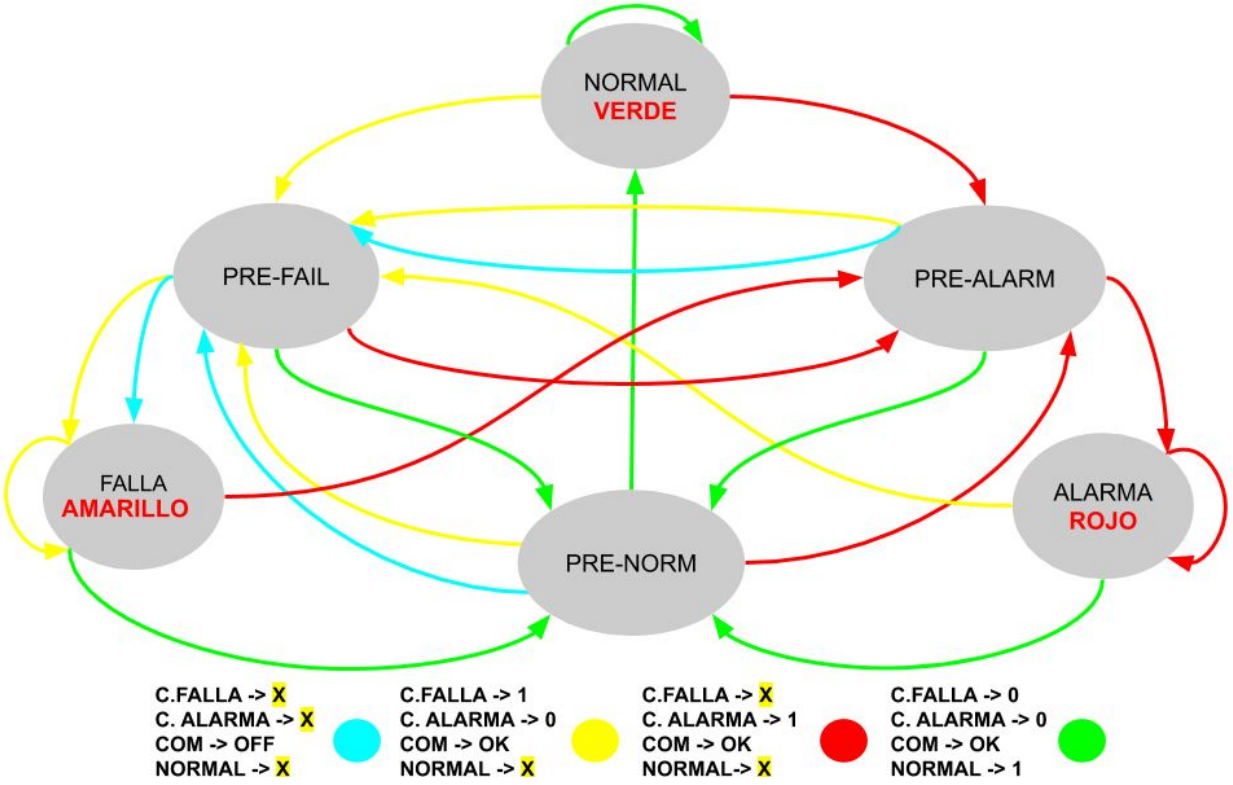
El usuario podrá modificar el estado por mensajes de UART al ejecutar el siguiente formato:

- Alarma → “al987” .
- Falla → “fa123”.
- Normal → “no456”.

Para realizar un cambio de estado, se deberá recibir al menos 2 veces consecutivas el código/evento que generó la transición, dentro de una ventana de tiempo de 3,5 segundos, en caso contrario nos trasladaremos a falla.

Periféricos (1 ó 2):
UART, GPIO

Diagrama de estado de MEF con breve descripción de cada estado.



Normal: No hay eventos de falla ni alarma presentes.

- Salida: Encender el led Verde.
- Transición:
  - Pre-Alarma: Se detectó una posible alarma.
  - Pre-Fail: Se detectó una posible falla.
  - Normal: Se mantiene el estado normal.

Falla: Existen eventos de falla, sin alarmas presentes.

- Salida: Encender el led Amarillo.
- Transición:
  - Pre-Alarm: Se detectó una posible alarma.
  - Pre-Norm: Se normaliza el estado.
  - Falla: Se mantiene el estado de falla.

Alarma: Existen eventos de alarma.

- Salida: Encender el led Rojo.
- Transición:
  - Pre-Fail: Se detectó una posible falla.
  - Pre-Norm: Se normaliza el estado.
  - Alarma: Se mantiene el estado de alarma.

Pre-Alarma: Etapa de transición.

- Salida: Ninguna.
- Transición:
  - Pre-Fail: Se detectó una posible una falla.
  - Pre-Norm: Se normaliza el estado.
  - Alarma: Se detectó una alarma.

Pre-Falla: Etapa de transición.

- Salida: Ninguna.
- Transición:
  - Pre-Alarm: Se detectó una posible alarma.
  - Pre-Norm: Se normaliza el estado.
  - Falla: Se detectó una falla.

Pre-Normal: Etapa de transición.

- Salida: Ninguna.
- Transición:
  - Pre-Fail: Se detectó una posible falla.
  - Pre-Alarm: Se detectó una posible alarma.
  - Normal: Se mantiene el estado normal.

Definir los módulos de software (archivos) que va implementar para cada periférico.

Primario_UART.c y Primario_UART.h para el manejo del periférico UART y recepción de los estados deseados por el usuario.

Primario_LEDS.c y Primario_LEDS.h para el manejo del periférico GPIO y el encendido de leds de la EDU-CIAA_NXP.

antirebote.c y antirebote.h para el antirebote de los pulsadores de la EDU-CIAA_NXP.

Primario.c y Primario.h para la lógica del dispositivo primario.

userTasks.c y userTasks.h definición de las tareas a ejecutarse por el sistema operativos.

main.c y para ejecutar el programa.

Definir los prototipos de las principales funciones públicas y privadas de cada módulo definido.

Primario_UART.c

- typedef struct{ uartMap_t Uart ; waitForReceiveStringOrTimeout_t waitText ; waitForReceiveStringOrTimeoutState_t waitTextState ; bool_t type ; uart_mode_t mode ; }uart_prim_t;
- char UARTRead(uart_prim_t * uprim); // Lectura de los estados del mensaje por recibir mediante el puerto UART hasta que se cumpla el tiempo máximo de espera o el mensaje patrón.
- void UARTUpdate(uart_prim_t * uprim); // Actualización del puerto y Lectura de los bytes recibidos.
- bool_t UARTInit(uart_prim_t * uprim,bool_t code,uartMap_t Uart) // Inicialización del puerto UART y definición del mensaje patrón.

Primario.c

- typedef enum{ NORMAL , ALARM, FAIL, PRENORMAL, PREALARM, PREFAIL} dprim_state_t;
- typedef struct{ tick_t timeout; delay_t delay; dprim_state_t state; antirebote_t boton1; antirebote_t boton2; antirebote_t boton3; uart_prim_t uart1; int8_t count; bool_t UARTFLAG; bool_t TEST_MODE; } dprimario_t;
- bool_t primInit(dprimario_t * primario); // Inicializa los parámetros del dispositivo primario.
- bool_t primControl(dprimario_t * primario); // Actualización de la máquina de estados para el control del dispositivo primario.

Primario_LEDS.c

- void turnOn(gpioMap_t lamp); // Encendido de Led Indicado.
- void turnOff(gpioMap_t lamp); // Apagado del Led indicado.

antirebote.c

- typedef enum{Init, UP, FALLING, DOWN, RISING} fsmstate_t;
- typedef struct{bool_t flag, gpioMap_t button, delay_t timing, fsmstate_t state} antirebote_t;
- (antirebote_t * pbutton, gpioMap_t boton); // Inicializa los parámetros de la MEFs de antirebote.
- void fsmUpdate(antirebote_t * pbutton); // Actualización de la máquina de estados para el control del antirebote.

userTasks.c

- void TaskInit(void* taskParmPtr); //Tareas de Inicialización.
- void Primario(void* taskParmPtr); //Tarea de ejecución de la lógica del dispositivo.
- void AllUpdates(void* taskParmPtr);// Tarea de ejecución de actualización de estados.
- void CurrentState(void* taskParmPtr);// Tarea de reporte de estado actual.
- void CurrentTmode(void* taskParmPtr); // Tarea de reporte de modo de prueba.
- void Test(void* taskParmPtr); // Tarea de ejecución de la lógica del modo de prueba.