Alumno: Gonzalo Vila

Plataforma embebida: STM32 NUCLEO-F429ZI

Aplicación:
La aplicación forma parte del trabajo práctico final del alumno. Este consiste en el desarrollo de un software para verificar prototipos de hardware antes de que sean entregados al cliente, quien programará el firmware.

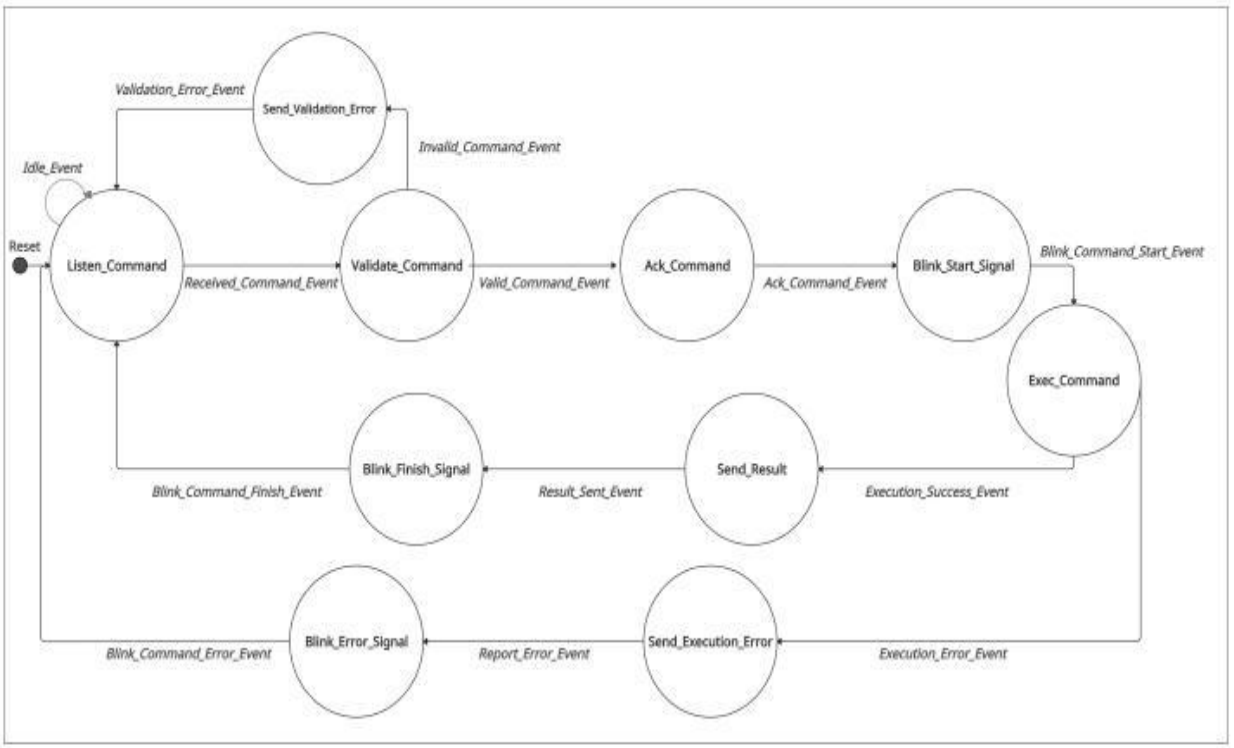
Para el trabajo final de esta materia, se desarrollará un módulo de software que se desplegará sobre el hardware embebido y permanecerá en modo de "escucha" hasta recibir un comando a través de la UART o detectar que el botón de usuario en la placa ha sido presionado.

El comando recibido desde la UART debe ser comparado contra una lista de instrucciones conocidas. De ser verificado, se debe notificar al usuario que la ejecución de un comando ha comenzado mediante el encendido de la luz verde de la placa. El resultado de la ejecución del comando debe retornar a una consola en la PC de destino a través de la UART. Al terminar se hace parpadear dos veces el led verde para luego apagarlo.

Los comandos disponibles retornarán información del hardware y el estado de la aplicación.
El botón de la placa actuará como un pulsador de emergencia que enviará un mensaje a través de la UART y hará parpadear tres veces todos los leds de la placa.

Periféricos (1 ó 2):
UART

Diagrama de estado de MEF



Breve descripción de cada estado

Listen_Command: Implementa un ciclo que escucha la llegada de comandos a través de la UART o la pulsación del botón de usuario en la placa.

Validate_Command: Verifica que el comando recibido se encuentre dentro de un listado de comandos válidos.

Send_Validation_Error: Envía un mensaje a la UART cuando el comando recibido no es válido.

Ack_Command: Envía un acuse de recibo del comando a través de la UART.

Blink_Start_Signal: Enciende el LED verde de la placa para notificar al usuario que la ejecución de un comando ha comenzado.

Exec_Command: Ejecuta, dependiendo del comando recibido, una de varias funciones de verificación del hardware.

Send_Result: Envía la salida del comando de verificación de hardware a través de la UART hacia la consola en la PC.

Blink_Finish_Signal: Parpadea tres veces el LED verde y luego lo apaga para señalizar que el comando ha terminado.

Send_Execution_Error: Envía a través de la UART el mensaje de error del comando.

Blink_Error_Signal: Parpadea cinco veces el LED rojo y luego lo apaga para señalizar que el comando ha fallado.

Módulos de software

EDTF_TESTSERVER.h: Archivo de cabecera que expone los prototipos de las funciones para inicializar y controlar el servidor de test.

EDTF_TESTSERVER.c: Contiene las funciones que inicializan el servidor de test y gestionan los cambios de estado de la máquina de estados.

EDTF_UART.h: Archivo de cabecera que expone los prototipos de las funciones para controlar el intercambio de datos a través de la UART.

EDTF_UART.c: Contiene las funciones que inicializan y gestionan la comunicación de datos a través de la UART. Abstrae el resto del código de los detalles de la comunicación con la consola de destino.

EDTF_LED.h: Archivo de cabecera que expone los prototipos de las funciones para inicializar y controlar los leds incluidos en el board.

EDTF_LED.c: Contiene las funciones que inicializan y gestionan los LEDS del board. Abstrae el resto del código de los detalles de su funcionamiento.

EDTF_BUTTON.h: Archivo de cabecera que expone los prototipos de las funciones para inicializar y controlar el *push button* incluido en el board.

EDTF_BUTTON.c: Contiene las funciones que inicializan y gestionan el *push button* del board. Abstrae el resto del código de los detalles de su funcionamiento. Implementa el algoritmo anti-rebote.

EDTF_TESTCOMMANDS.h: Es una librería que contiene los prototipos de los comandos que puede ejecutar el usuario. Se pretende desacoplar los *test* de la máquina de estados que los gestiona.

EDTF_TESTCOMMANDS.c: *Contiene las funciones que representan los test entre los que puede seleccionar el usuario.*

Prototipos de las principales funciones

EDTF_TESTSERVER.h

```
typedef enum {
	LISTEN_COMMAND,
	VALIDATE_COMMAND,
	ACK_COMMAND,
	BLINK_START_SIGNAL,
	EXEC_COMMAND,
	SEND_RESULT,
	BLINK_FINISH_SIGNAL,
	SEND_VALIDATION_ERROR,
	BLINK_ERROR_SIGNAL,
	SEND_EXECUTION_ERROR
}testServerStates_t;

typedef enum {
	IDLE_EVENT,
	RECEIVED_COMMAND_EVENT,
	VALID_COMMAND_EVENT,
	ACK_COMMAND_EVENT,
	BLINK_COMMAND_START_EVENT,
	EXECUTION_SUCCESS_EVENT,
	RESULT_SENT_EVENT,
	BLINK_COMMAND_FINISH_EVENT,
	VALIDATION_ERROR_EVENT,
	INVALID_COMMAND_EVENT,
	BLINK_COMMAND_ERROR_EVENT,
```

```
        REPORT_ERROR_EVENT,
        EXECUTION_ERROR_EVENT
}testServerEvents_t;

bool_t initTestServer ()
void testServerUpdate();
```

EDTF_BUTTON.h

```
bool_t debounceInit();
void debounceUpdate();
bool_t isButtonPressedEvent();
bool_t isButtonReleasedEvent();
uint32_t buttonPressedCount();
```

EDTF_UART.h:

```
bool_t initUart();
void uartSendResult(uint8_t * presult, uint16_t size);
```

EDTF_TESTCOMMANDS.h

```
testResult_t testButtonCommand ( );
testResult_t testPeripheralCommand ( );
```