Alumno: Storaccio Luis Sebastián

Plataforma embebida: **NÚCLEO F429ZI**

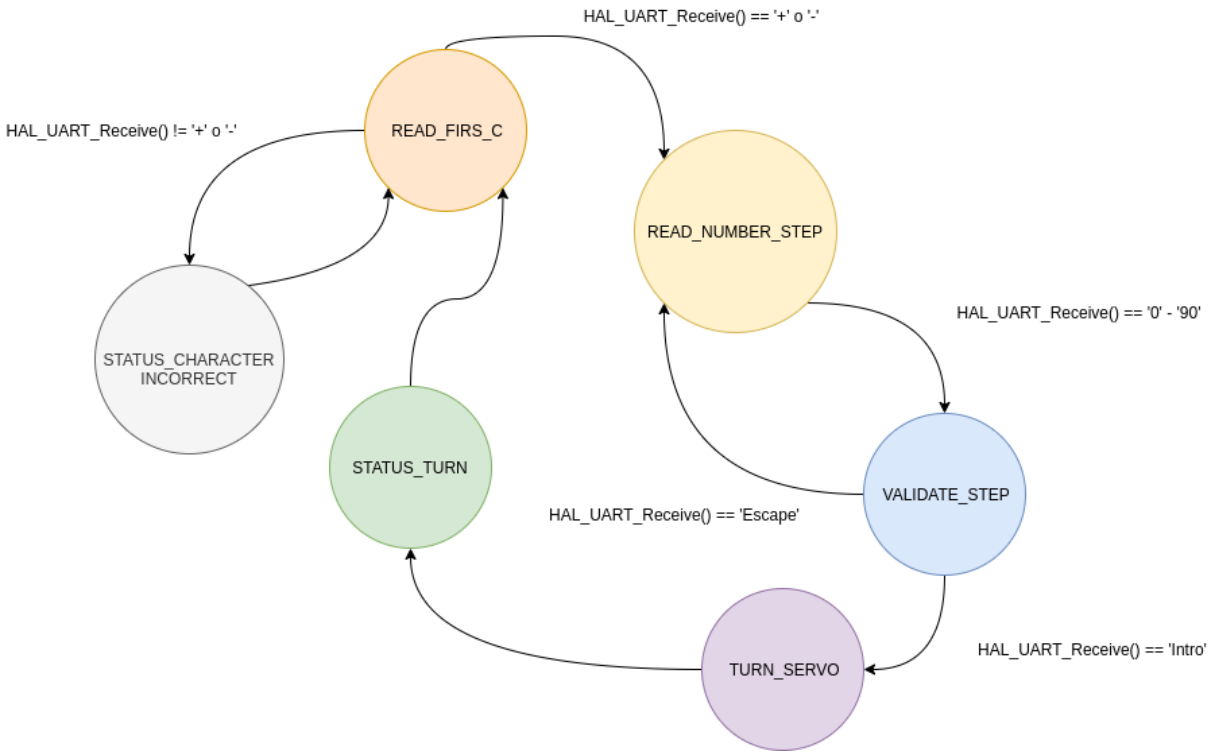
Aplicación:

Modulación de ancho de pulso (PWM) para mini servo SG90 . Los grados y el sentido de giro son ingresados por la terminal RS232 mediante un menú realizado con una MEF.

Periféricos:

UART, TMR2, GPIO.

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



Incluir una breve descripción de cada estado

READ_FIRST_C: es el estado inicial de la MEF y espera recibir un carácter por la terminal RS232 para seleccionar el sentido de giro. Con ‘+’ giro en sentido horario y con ‘-’ anti horario. En caso de ingresar otro carácter el estado siguiente es el STATUS_CHARTER_INCORRECT.

STATUS_CHARTER_INCORRECT: Informa al usuario que el carácter ingresado no es el esperado. Lleva a la MEF al estado inicial.

READ_NUMBER_STEP: este estado espera recibir un 2 carácteres por la terminal de ‘0’ a ‘9’ para indicar los grados a girar de 0 a 90°.

VALIDATE_STEP: este estado espera la confirmación del usuario de los carácteres ingresado anteriormente del valor de grados a girar. Presionando enter continúa al estado TURN_SERVO y escape para volver al estado READ_NUMBER_STEP.

TURN_SERVO: este estado realiza la conversión de los carácteres ingresados del valor de los grados a una variable entera. Luego convierte los valores de grados a milisegundos para el duty cycle de la salida PWM. Por último, inicializa el PWM.

STATUS_TURN: este estado informa al usuario que el servo se encuentra en movimiento y lleva la MEF al estado inicial.

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

API_delay.c y API_delay.h para generar retardos a tiempos definidos por el usuario en milisegundos.
API_uart.c y API_uart.h para inicializar el puerto y generar funciones para el envío y recepción de datos por la UART.
API_menu.c y API_menu.h para la realización del menú para la recepción de datos por la UART con una MEF y la generación del pulso para mover el servo.

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

API_delay.c

- **void delayInit(delay_t * delay, tick_t duration)** // Función que inicializa el temporizador. Recibe como parámetros un puntero a la estructura delay, con la información del temporizador actual, la duración y el estado del timer.
- **bool_t delayRead(delay_t *delay)** // Función que recibe como parámetro un puntero a la estructura del timer y devuelve en formato booleano si el tiempo transcurrió o no.
- **void delayWrite(delay_t * delay, tick_t duration)** // Función que recibe como parámetros un puntero a la estructura delay, y la duración en milisegundos del temporizador.

API_uart.c

- **UART_HandleTypeDef UartHandle;** // definición de una variable del tipo struct para la configuración de la UART.
- **bool_t uartInit()** // Función que inicializa la uart con la siguiente configuración:

```
UartHandle.Instance          = USARTx;
UartHandle.Init.BaudRate   = 9600;
UartHandle.Init.WordLength = UART_WORDLENGTH_8B;
UartHandle.Init.StopBits   = UART_STOPBITS_1;
UartHandle.Init.Parity         = UART_PARITY_ODD;
UartHandle.Init.HwFlowCtl  = UART_HWCONTROL_NONE;
UartHandle.Init.Mode         = UART_MODE_TX_RX;
UartHandle.Init.OverSampling = UART_OVERSAMPLING_16;
```

  La función devuelve true si la configuración se realizó con éxito.
- **void uartSendString(uint8_t *pstring,uint16_t size)** // Función que envía una cadena de carácteres por la terminal RS232, recibe como paŕamentros un puntero al string y el tamaño de la cadena.

- **uint8_t uartReceiveString(uint8_t *pData, uint16_t Size, uint32_t Timeout)** // Función que recibe un carácter por la terminal RS232, recibe como parámetros un puntero a la variable que almacenará el dato, el tamaño del dato recibido y el tiempo de espera del mismo.

API_menu.c

- **bool_t menuInit(void)** // Función que inicializa la MEF y asigna como primer estado menuState = READ_FIRS_C;
  Declaración de estados MEF:

```
typedef enum{
        READ_FIRS_C,
        READ_NUMBER_STEP,
        VALIDATE_STEP,
        TURN_SERVO,
        STATUS_TURN,
        STATUS_CHARACTER_INCORRECT,
} menu_t; }
```

- **void menuUpdate(uint16_t *dirP)** // Función que actualiza la MEF y los estados con el ingreso de los caracteres por la terminal RS232, recibe como puntero la dirección de la variable duty cycle.
- **static int CharToInt(uint8_t *data)** // Función que convierte los caracteres recibido por la terminal en una variable int, para determinar el ángulo de giro.

main.c

- **void SystemClock_Config(void)** // Función que configura el reloj del sistema.
- **static void MX_GPIO_Init(void)** // Función que configura la GPIO del microcontrolador
- **static void MX_TIM2_Init(uint16_t period)** // Función que configura los registros del timer para generar el pulso PWM.
- **void PWM_START(void)** // Función que inicializa el PWM .