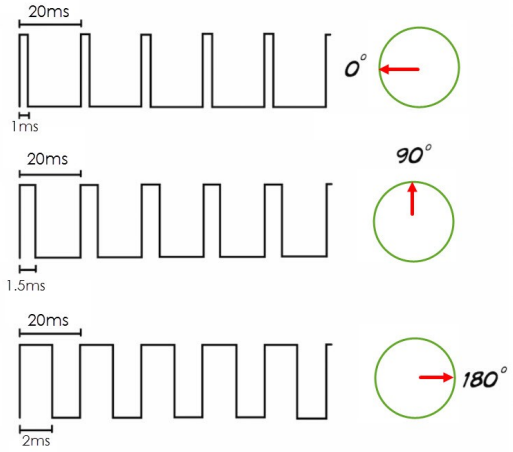
Trabajo final de PdM

Autor: Storaccio Luis

Modulación PWM para servomotor SG90

Se realizará un programa que genera un pulso PWM para el movimiento del servomotor SG90. Los grados y el sentido de giro son ingresados por la terminal RS232.



Periféricos utilizados y sus configuraciones

- **UART**: comunicación RS232 para que el usuario ingrese los grados y sentido de giro.
  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;
```

- **GPIO:** habilitación de indicadores luminosos LEDS.
  Configuración:

```
/* Configure the GPIO_LED pin */
GPIO_InitStruct.Pin = GPIO_PIN[Led];
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FAST;
```

Periféricos utilizados y sus configuraciones

- **TMR2** y **PWM:** Temporizador utilizado para generar la base de tiempo a 50 Hz para el PWM. El ancho del pulso lo modifica el usuario por la terminal RS232.

Configuraciones:

```
htim2.Instance = TIM2;
htim2.Init.Prescaler = 127;
htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
htim2.Init.Period = 13130;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
sConfigOC.OCMode = TIM_OCMODE_PWM1;
sConfigOC.Pulse = duty;
sConfigOC.OCPolarity = TIM_OCPOLARITY_HIGH;
sConfigOC.OCFastMode = TIM_OCFAST_DISABLE;
```

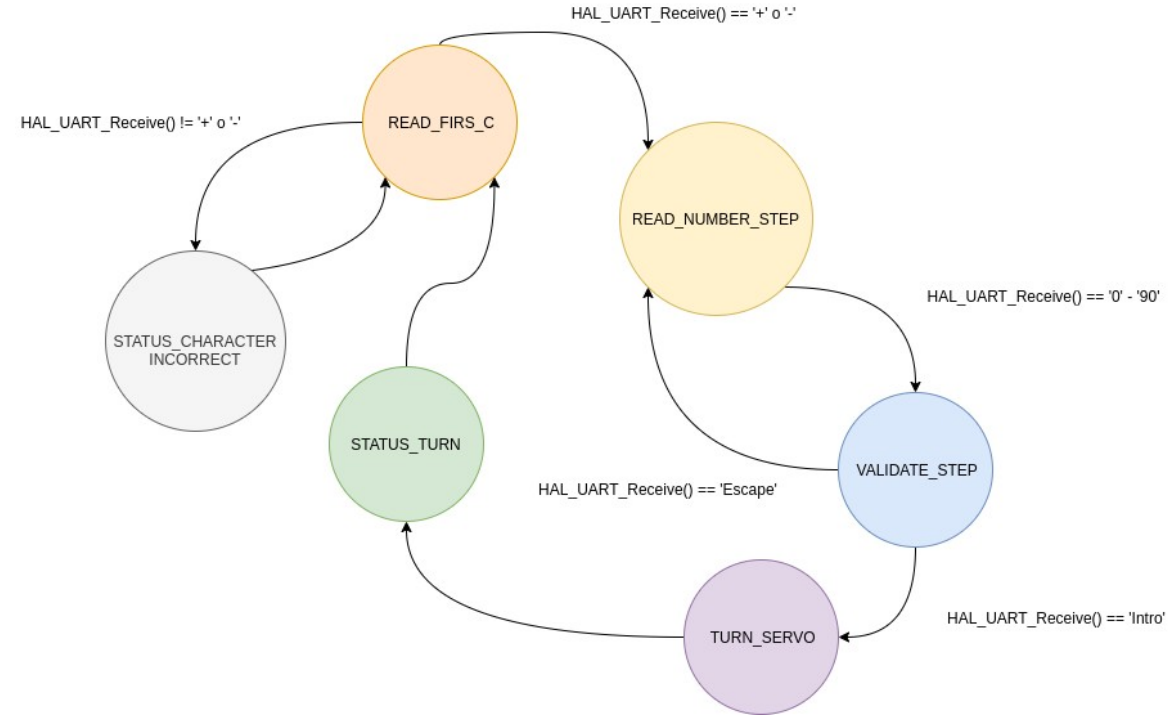
Modularización

- **API_delay.c y API_delay.h** para generar retardos no bloqueantes a tiempos definidos por el usuario en milisegundos.

- **API_uart.c y API_uart.h** para inicializar el puerto serie 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 provenientes de la terminal RS232 con una MEF. Además, se calcula el valor del ancho del pulso para la señal PWM.

Máquina de Estado Finita




```c
// Definición de estados MEF
typedef enum{
  READ_FIRS_C,
  READ_NUMBER_STEP,
  VALIDATE_STEP,
  TURN_SERVO,
  STATUS_TURN,
  STATUS_CHARACTER_INCORRECT,
} menu_t;
```

```c
// Función Inicializar MEF
bool_t menuInit(void)
{
delayInit(&delay,DELAY_TIME );
delayInit(&duty,DELAY_TIME );
menuState = READ_FIRS_C;
secuencia=false;
return true;
}
```

¡GRACIAS!