TRABAJO MICROS SED 2023/2024: MICROONDAS

Grupo 6

Realizado por:

Javier Robinat Cuiñas 55430

Fernando Marín Raez 54724

Álvaro Rico García 55425

Repositorio Github: https://github.com/Alvarorico/Microondas/tree/main

Índice:

Introducción

En este proyecto, se abordará la simulación del funcionamiento de un microondas mediante el uso del entorno de desarrollo STM32. La simulación se basará en la implementación de un código en el lenguaje C, haciendo uso de la biblioteca de abstracción de hardware (HAL).

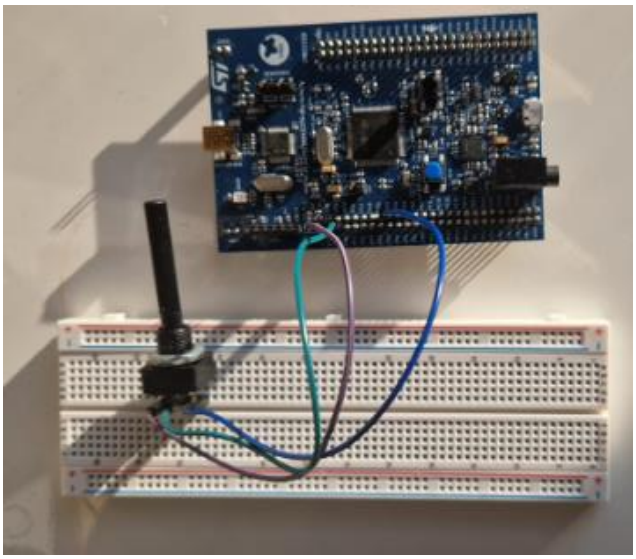
La idea principal es emular las funciones básicas de un microondas, como el control de tiempo, la activación del calentamiento y la interacción con el usuario a través de un botón.

El programa utiliza un potenciómetro como un convertidor analógico a digital (ADC) para leer un valor entre 0 y 255, que representa la cantidad de segundos que se configurarán en el temporizador del microondas.

Cuando se pulsa el botón de la placa, el programa inicia un temporizador utilizando el valor leído del potenciómetro como el tiempo de cuenta atrás.

Durante la cuenta atrás, el temporizador se decrementa y, cuando llega a cero, se enciende un LED para indicar que el tiempo ha acabado.

Este trabajo explorará cómo diseñar y simular el comportamiento de un microondas, integrando el hardware y el software.



Componentes

Potenciómetro:

El potenciómetro escogido es de tipo B100k, donde la B representa el tipo de curva resistiva y 100k indica una resistencia.

El potenciómetro es de tipo rotativo y consta de tres terminales. Al aplicar una tensión entre los terminales externos, la resistencia el terminal central y uno de los extremos, varía según la posición del contacto.

Configuración de pines

PA0	Botón por interrupción
PA1	Temporizador por interrupción
PA2	Potenciómetro (ADC)
PD12	LED verde
PD13	LED naranja

Cabe destacar que el potenciómetro está conectado a una protoboard.

Implementación del código

Encendido led:

El código proporcionado es una interrupción externa (“HAL_GPIO_EXTI_Callback”). Este callback se activa cuando se pulsa el botón (GPIO_PIN_0). Cuando ocurre esta interrupción, se establece la variable “buttonPressed” en 1 y se enciende un LED que simula el encendido del microondas. El LED en cuestión está en el pin GPIO_PIN_12 del puerto D (GPIOD). Por lo tanto, la función representa la lógica asociada a la pulsación de un botón que enciende un indicador visual (el LED) para simular el encendido del microondas. Al mismo tiempo el temporizador se activa dentro de la interrupción.

```
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if (GPIO_Pin == GPIO_PIN_0)
  {
      //buttonPressed = 1;
      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
      __HAL_TIM_CLEAR_FLAG(&htim2, TIM_IT_UPDATE);
      HAL_TIM_Base_Start_IT(&htim2);

  }
}
```

Temporizador:

El callback del temporizador ("HAL_TIM_PeriodElapsedCallback") desempeña un papel importante en la simulación del microondas. Cuando el temporizador TIM2 alcanza el final de su periodo, se activa esta interrupción. En respuesta, se marca la variable time_end como 1, indicando la finalización del periodo. Además, se enciende el LED asociado al pin GPIO_PIN_13 en el puerto D (GPIOD), proporcionando una señal visual de que ha transcurrido el tiempo establecido.

Finalmente, se detiene el propio temporizador TIM2 y se desactivan las interrupciones de tiempo correspondientes, consolidando así la simulación del fin del ciclo de operación del microondas.

```
66 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
67 {
68   if (htim->Instance == TIM2)
69   {
70       time_end = 1;
71       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
72       HAL_TIM_Base_Stop_IT(&htim2);
73
74   }
75 }
76 void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
77 {
78   if (GPIO_Pin == GPIO_PIN_0)
79   {
80       buttonPressed = 1;
81       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
82       HAL_TIM_Base_Start_IT(&htim2);
83       __HAL_TIM_CLEAR_FLAG(&htim2, TIM_IT_UPDATE);
84   }
85 }
```

$$T = \left(\frac{fclk}{(preescaler + 1) * (period + 1)}\right)^{-1}$$

T=5 s

```
htim2.Instance = TIM2;
htim2.Init.Prescaler = 8000-1;
htim2.Init.CounterMode = TIM_COUNTERMODE_DOWN;
htim2.Init.Period = 5000-1;
htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_DISABLE;
```

Potenciómetro:

En este fragmento de código, se implementa un bucle principal que continuamente lee el valor del potenciómetro mediante un convertidor analógico a digital (ADC).

Durante cada iteración del bucle, se inicia la conversión ADC con “HAL_ADC_Start”, se espera a que la conversión se complete mediante “HAL_ADC_PollForConversion”, y el valor convertido se recupera y almacena en la variable “ADC_val”. Posteriormente, se detiene el ADC con “HAL_ADC_Stop”.

Este proceso se repite de forma continua, permitiendo la lectura y actualización constante del valor del potenciómetro en el programa.

```
uint32_t ADC_val;
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    // Leer el valor del potenciómetro
    HAL_ADC_Start(&hadc1);

      if(HAL_ADC_PollForConversion(&hadc1, HAL_MAX_DELAY) == HAL_OK){
          ADC_val = HAL_ADC_GetValue(&hadc1);
      }
     HAL_ADC_Stop(&hadc1);
```

En el siguiente código se ha añadido la función HAL_TIM_SET_COUNTER en la cual se carga el valor del potenciómetro, leído en el bucle while descrito anteriormente, en el temporizador. Dicho valor se multiplica por 100 para un correcto funcionamiento.

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
  if (htim->Instance == TIM2)
  {
      //time_end = 1;
      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
      HAL_TIM_Base_Stop_IT(&htim2);

  }
}
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
  if (GPIO_Pin == GPIO_PIN_0)
  {
      //buttonPressed = 1;
      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
      __HAL_TIM_CLEAR_FLAG(&htim2, TIM_IT_UPDATE);
      __HAL_TIM_SET_COUNTER(&htim2, ADC_val*100);
      HAL_TIM_Base_Start_IT(&htim2);

  }
```

Una vez probado el código en la placa, se aprecia que, al aumentar el valor del potenciómetro, aumenta el tiempo que tarda en encenderse el LED (naranja)

configurado para marcar el final del temporizador. Los tiempos máximo y mínimo son 24 y 6 segundos respectivamente.