ELECTRÓNICA DIGITAL III

Trabajo Práctico Final:

Sistema de Radar con STM32

Alumnos:

Cechich, Ignacio

Rodriguez, Mateo

Trachta, Agustin

Profesor:

Ing Gallardo, Fernando

Año 2024

Introducción

Este informe describe el desarrollo de un sistema de radar utilizando un microcontrolador STM32, un servomotor, un sensor ultrasónico y un potenciómetro para el control del ángulo del radar. El sistema mide distancias de objetos y transmite los datos a una computadora para su visualización. Se explicarán las configuraciones y el flujo de trabajo completo del proyecto, desde la inicialización hasta la transmisión de datos.

Objetivo del Proyecto

El objetivo principal es implementar un sistema que utilice un sensor ultrasónico para medir la distancia de objetos mientras se controla el ángulo de escaneo con un servomotor. La información del ángulo y la distancia se transmite a una computadora para su análisis y visualización. Este sistema simula el comportamiento básico de un radar, proporcionando información continua de la distancia de los objetos detectados.

Componentes Utilizados

- **STM32F103C8T6**: Microcontrolador que gestiona la lógica de control, la adquisición de datos y la comunicación.
- **Sensor Ultrasónico HC-SR04**: Utilizado para medir la distancia a objetos cercanos.
- **Servomotor**: Controla el ángulo de rotación del radar para barrer diferentes direcciones.
- **Potenciómetro**: Permite ajustar el ángulo del servomotor manualmente.
- **DAC y Buzzer** : Generan un tono proporcional a la distancia detectada.
- **UART (comunicación serie)**: Se utiliza para transmitir los datos a la PC para visualización.

Flujo de Trabajo del Sistema

1. **Inicialización**: Al encender el sistema, se configuran todos los periféricos necesarios:
    - **Sistema de reloj**: Se configura para proporcionar frecuencias adecuadas a los buses y periféricos.
    - **GPIO**: Configura los pines para la entrada/salida del sensor ultrasónico, el control del servomotor, y la UART.
    - **ADC**: Configura la conversión del valor del potenciómetro para controlar el ángulo del servomotor.
    - **UART**: Configura la comunicación con la PC para transmitir los resultados.
    - **TIM1 y TIM2**: Configura los temporizadores; `TIM1` para medir la señal del sensor ultrasónico y `TIM2` para generar la señal PWM que controla el servomotor.
2. **Control del Servomotor**:
    - **Lectura del Potenciómetro**: El ADC lee el valor del potenciómetro que indica el ángulo deseado del servomotor.

- **Filtrado del Valor ADC**: Se utiliza un filtro de promedio móvil para reducir el ruido de la señal del potenciómetro y obtener un valor más estable.
- **Mapeo del Valor ADC a PWM**: El valor del ADC se mapea a un rango adecuado para controlar el PWM del servomotor (600 a 2400), que corresponde a un ángulo entre 0° y 180°.

Para controlar el servomotor, utilizamos un valor PWM que se ajusta entre 600 y 2400 (que representa el valor del registro de comparación del temporizador).

Este rango se corresponde con un ciclo de trabajo que varía de aproximadamente 3% a 12% del periodo total de 20 ms, típico para el control de servomotores estándar.

El valor del ADC, que varía de 0 a 4095 (resolución de 12 bits), se mapea a este rango de PWM para obtener un control suave y preciso del ángulo del servomotor.

El cálculo se realiza mediante la siguiente fórmula de mapeo:

$$pwm_{val} = \frac{adc_{val}}{4095} * (2400 - 600) + 600$$

Esto asegura que el valor PWM esté en el rango adecuado para controlar el servomotor de 0° a 180°.

- **Control del Servomotor**: El valor mapeado se utiliza para ajustar el PWM en `TIM2`, lo que mueve el servomotor al ángulo deseado.

3. **Medición de Distancia con el Sensor Ultrasónico**:
   - Se genera un pulso de 10 microsegundos en el pin `TRIG` del sensor ultrasónico, iniciando la medición.
   - El pin `ECHO` recibe la respuesta del sensor cuando detecta un objeto, y `TIM1` mide el tiempo que tarda el pulso en regresar.
   - La distancia se calcula con la fórmula:
     “Distancia = (Tiempo de eco * Velocidad del sonido) / 2”.
4. **Generacion de Sonido Proporcional a la Distancia**
   - Uso del DAC controlado por DMA para generar una señal de onda senoidal que varía su frecuencia según la distancia detectada.
   - La frecuencia del buzzer es inversamente proporcional a la distancia medida, alertando al usuario sobre objetos cercanos.
5. **Envío de Datos a la PC**:
   - Los datos del ángulo y la distancia se formatean en un mensaje del tipo `“<ángulo>,<distancia>.”`.
   - Este mensaje se envía a la PC a través del puerto UART.
6. **Visualización de Datos**:
   - Los datos recibidos por la PC se pueden usar para visualizar el funcionamiento del radar y los objetos detectados.

Uso del DMA

El DMA (Acceso Directo a Memoria) se utiliza para optimizar las transferencias de datos, liberando a la CPU para otras tareas. En este proyecto, el DMA se aplica de las siguientes formas:

DMA para ADC

- Los datos convertidos por el ADC son transferidos automáticamente a un buffer en memoria, sin intervención de la CPU. Esto permite lecturas continuas del potenciómetro de forma eficiente, mejorando la precisión y reduciendo la carga de trabajo.

DMA para DAC y Generación de Sonido

- Se utiliza el DMA para transferir automáticamente una tabla precargada de valores de una onda senoidal desde la memoria al DAC. Esta tabla representa un ciclo completo de la onda, que es enviada repetidamente al DAC para generar una señal continua.
- La frecuencia de la onda senoidal, y por ende del tono generado por el buzzer, se ajusta dinámicamente en función de la distancia medida por el sensor ultrasónico.
- El proceso es altamente eficiente, ya que el microcontrolador solo actualiza los parámetros del temporizador que controla la frecuencia del DAC, mientras el DMA maneja las transferencias de datos.

Mejoras Implementadas

1. **Filtro de Promedio Móvil**

   Se implementó un filtro de promedio móvil con los últimos 5 valores del ADC para reducir fluctuaciones. Esto estabilizó el movimiento del servomotor, especialmente en ángulos críticos como los 90**°.**

2. **Generación de Sonido Proporcional a la Distancia**

   Se utilizó el DAC para generar un tono que varía según la distancia detectada. El tono es más agudo cuando el objeto está cerca y más grave cuando está lejos.

   La implementación usa una tabla de valores senoidales precargada que se envía al DAC mediante DMA. El temporizador asociado ajusta la frecuencia de la señal, logrando variaciones dinámicas en el sonido del buzzer sin intervención directa de la CPU.

   Esto proporciona una representación auditiva intuitiva de la distancia, lo que resulta útil en aplicaciones donde la visualización no es suficiente.

A continuacion se detalla el codigo implementado para este apartado.

Se declararon los handlers de las estructuras y se definieron los puertos que se van a utilizar.

```
DAC_HandleTypeDef hdac1;
DMA_HandleTypeDef hdma_dac1_ch1;
```

```
#define BUZZER_PIN GPIO_PIN_4
#define BUZZER_PORT GPIOA
```

Se inicializo el DAC y se definio el canal a utilizar.

```
static void MX_DAC1_Init(void);
```

```
//Iniciar DAC para el buzzer
HAL_DAC_Start(&hdac1 , DAC_CHANNEL_1);
```

Mapeo de la distancia en 8 bits.

```
//Activar el buzzer en funcion de la distancia
uint8_t dac_value = (uint8_t)map_float(Distance, 0, 200, 0, 255); // Rango de 0 a 200 cm mapeado a 8 bits
HAL_DAC_SetValue(&hdac1 , DAC_CHANNEL_1 , DAC_ALING_8B_R , dac_value);//Aling 8 bit right
```

Configuración del pin elegido como salida del DAC.

```
//Configurar el pin DAC(PA4) como salida analógica
GPIO_InitStruct.GPIO_Pin = GPIO_Pin_4;
GPIO_InitStruct.GPIO_Mode = GPIO_Mode_AIN; // Modo Analógico
GPIO_Init(GPIOA, &GPIO_InitStruct);
```

Configuración del reloj y el canal del DAC.

```
//Configuracion DAC
static void MX_DAC1_Init(void){
    DAC_ChannelConfTypeDef sConfig = {0};

    //Habilitar reloj de DAC
  __HAL_RCC_DAC_CLK_ENABLE();

    hdac.instance = DAC;

    if (HAL_DAC_Init(&hadc1) != HAL_OK) {
    Error_Handler();
    }
    sConfig.DAC_OutputBuffer = DAC_OUTPUTBUFFER_ENABLE;
    HAL_DAC_ConfigChannel(&hdac, &sConfig, DAC_CHANNEL_1);  // Canal 1
}
```

Implementación de DMA junto al DAC.

```
/*Configuracion de DMA para dac*/
hdma_dac1.Instance = DMA_Channel2;
hdma_dac1.Init.Direction = DMA_MEMORY_TO_PERIPH;
hdma_dac1.Init.PeriphInc = DMA_PINC_DISABLE;
hdma_dac1.Init.MemInc = DMA_MINC_DISABLE;
hdma_dac1.Init.PeriphDataAlignmet = DMA_PDATAALIGN_HALFWORD;
hdma_dac1.Init.MemDataAlignmet = DMA_MDATAALIGN_HALFWORD;
hdma_dac1.Init.Mode = DMA_CIRCULAR;
hdma_dac1.Init.Priority = DMA_PRIORITY_LOW;

if (HAL_DMA_Init(&hdma_dac1) != HAL_OK) {
  Error_Handler();
}
__HAL_LINKDMA(&hdac, DMA_Handle1, hdma_dac1);
```

3. **Optimización del PWM para el Servomotor**

Se ajustaron el prescaler y el periodo del temporizador TIM2 para lograr un ciclo PWM preciso (20 ms, 50 Hz). Esto permitió un control suave y fluido del servomotor, eliminando temblores y microcortes.

Problemas y Soluciones

- **Temblores del Servomotor**:

Se observó que el servomotor temblaba especialmente al llegar a los 90 grados. Esto se debió a pequeñas variaciones en la lectura del ADC.

Fueron solucionados con un filtro de promedio móvil para estabilizar las lecturas del ADC.

- **Microcortes en el Movimiento del Servomotor**:

Se optimizó el valor del prescaler y la resolución del PWM para garantizar un movimiento más fluido y preciso.

La fórmula utilizada para determinar el valor del prescaler es:

$$frec_{TIM2} = \frac{frec\ del\ reloj\ principal}{PR+1}$$

Ajustar el prescaler y el valor de periodo permite aumentar la resolución del control PWM, lo que se traduce en un movimiento del servomotor más suave y preciso, evitando microcortes y temblores.

Conclusión

Este proyecto ha demostrado la capacidad de integrar múltiples componentes para crear un sistema de radar básico con un microcontrolador STM32. Utilizando el sensor ultrasónico para medir distancias y un servomotor para controlar el ángulo, el sistema proporciona datos en tiempo real a la PC para visualización. La implementación de filtros y ajustes precisos en el PWM ha mejorado considerablemente la estabilidad y precisión del sistema. Este proyecto es una excelente base para aprender sobre el control de motores, la adquisición de datos con sensores y la comunicación serie.

Anexo

Enlace al repositorio que contiene el código corregido.

https://github.com/Ignacio3000/ED_III/blob/44937a11d1357c5452d0b80b23c84ed69b9e9a66/TP_FINAL_EDIII.c

Imagenes del trabajo: