### Introducción

#### Contexto del proyecto

Este proyecto de brazo robótico se inspira en una versión inicial realizada en la escuela utilizando Arduino. Con los conocimientos adquiridos en la universidad, el objetivo es mejorar y expandir esa idea. El brazo está diseñado para ejecutar comandos en tiempo real y almacenar patrones de movimiento aprendidos para su reproducción en bucle. Este modelo de brazo robótico a pequeña escala se controla con un PSoC, aprovechando sus periféricos y capacidades de procesamiento para lograr los movimientos y funcionalidades requeridas.

#### Objetivo general

Desarrollar un brazo robótico controlado mediante PSoC, integrando una interfaz de usuario y almacenamiento de movimientos para permitir tanto el seguimiento en tiempo real de comandos como la repetición de secuencias previamente aprendidas.

#### Objetivos específicos

1. Crear un brazo con movimientos en base, codo, muñeca y pinza.

2. Diseñar una interfaz que permita al usuario ingresar comandos.

3. Incorporar un display y otros medios visuales que comuniquen al usuario el estado del sistema.

4. Configurar y utilizar los periféricos del PSoC para lograr una comunicación eficiente entre las partes del sistema.

5. Implementar una funcionalidad de almacenamiento de patrones de movimiento que permita al brazo aprender y reproducir secuencias en bucle.

6. Ajustar el control del brazo para que opere en la zona lineal de los potenciómetros, maximizando la precisión de los movimientos.

7. Almacenar los movimientos aprendidos en la EEPROM, evitando que se pierdan en caso de una interrupción de energía.

#### Justificación

La robótica tiene aplicaciones cruciales en la automatización de tareas repetitivas y precisas, así como en la investigación y educación. Este proyecto de brazo robótico permite experimentar con el control de servomecanismos, los cuales simplifican la implementación en comparación con otras opciones. La capacidad de aprender y reproducir movimientos emula bien un proceso de automatización, mientras que el modo de seguimiento en tiempo real simula entornos de robótica asistencial. Esto permite profundizar en los principios básicos de la robótica, a la vez que mejora las habilidades prácticas en el diseño y programación de sistemas embebidos.

.

### Análisis del sistema

#### Partes en las que se divide el sistema y su funcionalidad

1. \*\*Brazo robótico electromecánico\*\*

Ejecuta los movimientos mediante la rotación controlada de los motores en cada articulación (base, codo, muñeca y pinza), permitiendo que el brazo alcance las posiciones deseadas.

2. \*\*Interfaz de usuario\*\*

Permite al usuario ingresar comandos y visualizar mensajes del sistema, facilitando una interacción intuitiva y en tiempo real para configurar y controlar el brazo robótico.

3. \*\*Sistema de ingreso de posiciones de los motores\*\*

Recibe las posiciones específicas a las que deben moverse los motores, permitiendo que el usuario controle con precisión los movimientos del brazo.

4. \*\*Medio de almacenamiento\*\*

Guarda de forma segura los patrones de movimiento aprendidos en memoria no volátil, para que no se pierdan en caso de apagado, permitiendo su reproducción en cualquier momento.

#### Relación entre las partes

Todas las partes están conectadas y coordinadas a través del PSoC, que actúa como controlador central. La interfaz envía comandos al PSoC, que los interpreta y gestiona, asegurando que los motores se ajusten según las posiciones deseadas y que los movimientos se guarden para futuras ejecuciones.

**Diseño del sistema**

**Diseño de hardware**

El sistema del brazo robótico combina diversos componentes electrónicos y mecánicos para garantizar su funcionalidad. Los elementos principales son:

1. **Control de movimiento**  
   El movimiento del brazo se logra mediante servomotores, los cuales se controlan con pulsos PWM generados por el PSoC. Este enfoque permite una integración eficiente y precisa gracias a las capacidades nativas de PWM del microcontrolador.
2. **Interfaz de usuario**
   * **LCD RGB**: Se utiliza una pantalla LCD RGB para mostrar instrucciones y estados del sistema. La comunicación entre la LCD y el PSoC se realiza mediante el protocolo I2C.
   * **OLED**: Complementa al LCD con animaciones que representan visualmente el estado actual del sistema o indican errores mediante bitmaps almacenados en una tarjeta SD. Para su control, se utiliza un Arduino Mega conectado a la OLED mediante I2C y a la SD mediante SPI.
   * **Teclado matricial**: Permite ingresar comandos al sistema. El teclado incluye diodos para el barrido y se conecta al PSoC.
3. **Sensores y acondicionamiento de señal**  
   Los potenciómetros lineales, usados para medir los ángulos de las articulaciones, están conectados al ADC-SAR del PSoC a través de un MUX. Antes del MUX, se incluye un filtro pasivo pasa bajas para reducir el ruido y un amplificador operacional en configuración buffer para minimizar errores relacionados con la alta impedancia de entrada del ADC.
4. **Sincronización y control**  
   El sistema utiliza timers y registros de estado para sincronizar las acciones. Las interrupciones se emplean para tareas prioritarias, mientras que los registros sticky permiten manejar eventos no prioritarios de manera eficiente.

**Diseño de software**

El diseño de software se organiza en módulos para cada funcionalidad:

1. **Comunicación entre PSoC y Arduino**  
   El PSoC envía instrucciones al Arduino mediante UART utilizando un protocolo de confirmación:
   * El PSoC envía un mensaje y activa un temporizador.
   * El Arduino responde con el mismo mensaje.
   * Si el PSoC no recibe o detecta errores en la respuesta, reenvía el mensaje hasta que la transmisión sea exitosa o alcance un límite de intentos.
2. **Interfaz de usuario**
   * El teclado se maneja mediante interrupciones periódicas controladas por un temporizador.
   * La LCD RGB utiliza una versión adaptada de la librería Adafruit, mostrando información del estado y respondiendo a las entradas del usuario.
   * La OLED, controlada por el Arduino, muestra animaciones almacenadas en formato bitmap y gráficas en tiempo real, como barras de progreso basadas en las posiciones de los potenciómetros.
3. **Control de movimiento**  
   Los datos de los potenciómetros se ajustan para operar solo en su zona lineal. Los valores son procesados por el PSoC para generar los pulsos PWM necesarios para mover los servomotores.
   * Se utiliza una curva de ajuste con umbrales para evitar la zona no lineal.
   * Opcionalmente, se aplica un filtro de media móvil con tres niveles configurables.
4. **Estados del sistema**  
   El sistema opera bajo una máquina de estados con transiciones controladas por las entradas del usuario:
   * **START**: Inicialización y reinicio de componentes.
   * **BIENVENIDA**: Espera hasta que el sistema esté listo.
   * **MENU**: Permite al usuario activar o desactivar funciones, cambiar configuraciones, o elegir modos de operación.
   * **MODO\_SEGUIR**: Control en tiempo real de los motores según los potenciómetros.
   * **MODO\_GUARDAR**: Guarda posiciones en un vector con muestras periódicas o manuales.
   * **MODO\_LEER**: Ejecuta en bucle uno de los vectores guardados.
   * **MODO\_GUARDAR\_EEPROM**: Permite cargar o guardar datos en la EEPROM.
   * **MODO\_LEER\_SELECCION** y **MODO\_GUARDAR\_SELECCION**: Selección del vector a leer o guardar.
   * **MODO\_GUARDAR\_SELECCION\_:** Se elije si guardar los puntos con muestreo periódico o aperiódico mediante la tecla ENT.
   * **REINICIAR**: Reinicia el sistema.

El flujo principal del software combina la máquina de estados con tareas periódicas y priorizadas mediante timers e interrupciones.