Fase 2: Prótesis de mano*

Rodrigo Sosa Aquino, 202012337,[1] Cindy Melissa, Gatica Arriola, 201709692,[1] and Stheeven Adonías, Coc Chán, 201700519[1]

[1] *Facultad de Ingeniería, Universidad de San Carlos, Departamento de laboratorios de Electrónica, Laboratorio de Electrónica 3, segundo semestre 2024.*

Una prótesis de mano es un dispositivo médico diseñado para reemplazar parcial o totalmente la funcionalidad de una mano perdida, permitiendo al usuario recuperar ciertas habilidades motoras esenciales. Estas prótesis pueden ser pasivas, proporcionando una apariencia natural, o activas, utilizando tecnología mecánica o electrónica para permitir movimientos controlados, como el agarre y la manipulación de objetos. Hoy en día, las prótesis de mano son cruciales para mejorar la calidad de vida de personas con amputaciones, facilitando su reintegración social y laboral. Los avances en materiales y tecnologías, como la robótica y la interfaz cerebro-computadora, han incrementado la efectividad y personalización de estas prótesis, destacando su importancia en la medicina moderna.

I. CIRCUITOS PROPUESTOS

A. Conexión sensor muscular a la tiva TM4C123GH6PM

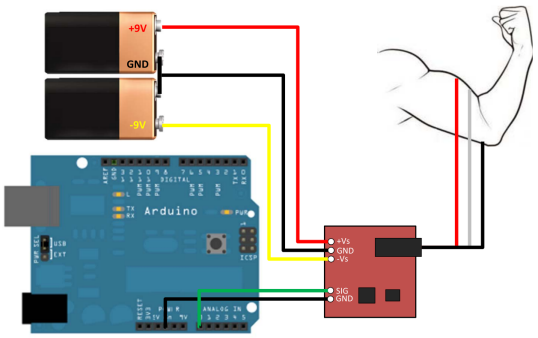


Figura 1: Conexión sensor a la tiva

II. LISTA DE IDEAS

* Utilizar servomotores para el movimiento de los dedos de la prótesis.
* Usar un módulo adicional para la lectura de los sensores musculares.
* Utilizar hilo grueso para conectarlo a cada uno de los dedos.
* Usar pajillas para pasar el hilo grueso y evitar que se mueva a otros lugares.

III. MATERIALES A UTILIZAR

1. *Sensores musculares*

La medición de la actividad muscular mediante la detección de su potencial eléctrico, conocida como electromiografía. Este sensor mide la actividad eléctrica filtrada y rectificada de un músculo; emitiendo desde 0 V hasta Vs V dependiendo de la cantidad de actividad en el músculo seleccionado, donde Vs significa el voltaje de la fuente de poder.

Detecta señales, amplifica y procesa la actividad eléctrica de un músculo y la convierte en una señal analógica simple que puede ser leída fácilmente por cualquier microcontrolador con un convertidor analógico a digital (ADC)

Figura 2: Sensor múscular

A. Microcontrolador Tiva TM4C123GH6PM

Recibe las señales procesadas. Posteriormente interpreta y envía señales a los servomotores.

* Práctica de eléctrónica 3

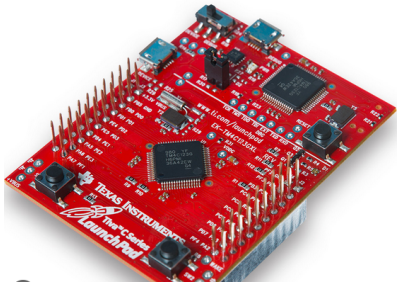


Figura 3: Tiva TM4C123GH6PM

B. Servomotores

Actuador rotativo o lineal que permite lograr un control preciso en cuanto a posición angular, aceleración y velocidad del eje. Permite cerrar o abrir los dedos de la prótesis de mano.

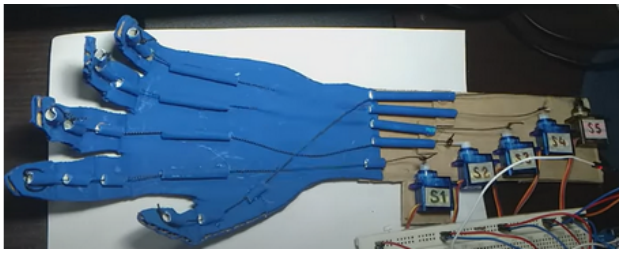


Figura 4: servomotor

C. Baterías

Fuente de alimentación para los sensores. Valor depende del tipo de sensor. Especificado en la datasheet.

Figura 5: baterias

D. Calibración de los sensores y valores de referencia

- Limpiar completamente el área deseada con jabón para eliminar la suciedad y el aceite
- Colocar los electrodos en los conectores a presión del sensor
- Colocar el sensor en el músculo deseado
  - Después de determinar qué grupo muscular se quiere tratar (p. ej., bíceps, antebrazo, pantorrilla), limpiar bien la piel
  - Coloca el sensor de manera que uno de los electrodos conectados esté en el medio del cuerpo muscular
  - El otro electrodo debe estar alineado en la dirección de la longitud del músculo
  - Colocar el electrodo de referencia en una parte ósea o muscular no adyacente de tu cuerpo cerca del músculo objetivo
  - Conéctar a una placa de desarrollo Tiva.

E. Posicionamiento de los sensores musculares

La posición y la orientación de los electrodos del sensor muscular tienen un gran efecto en la intensidad de la señal. Los electrodos deben colocarse en el centro del cuerpo muscular y deben estar alineados con la orientación de las fibras musculares. Colocar el sensor en otras ubicaciones reducirá la intensidad y la calidad de la señal del sensor debido a una reducción del número de unidades motoras medidas y a la interferencia atribuida a la diafonía.

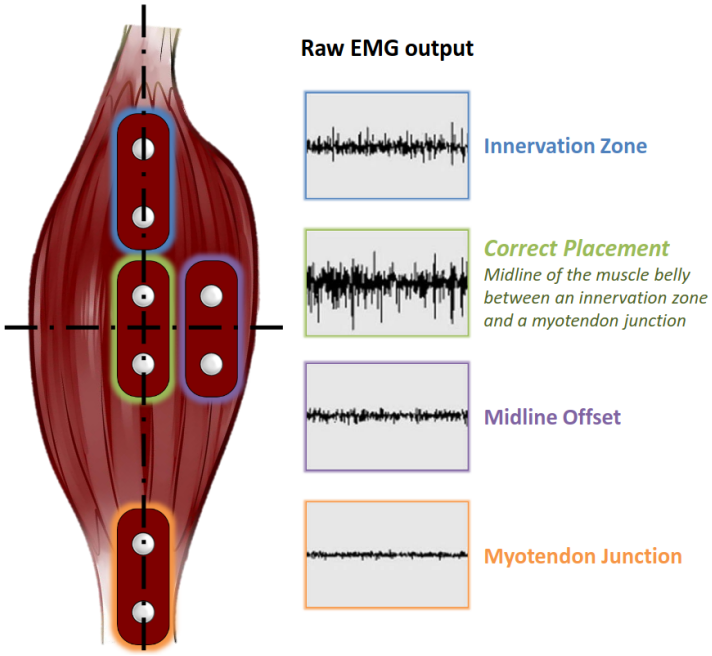


Figura 6: Posicionamiento correcto del sensor

IV. DIAGRAMA DE BLOQUES PROPIO

A. Diagrama de bloques propuesto

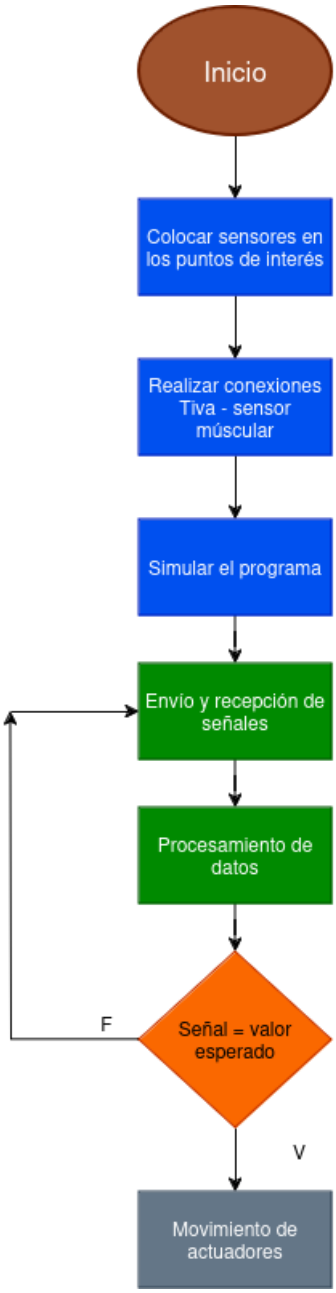


Figura 7: Diagrama de bloques

V. DUDAS

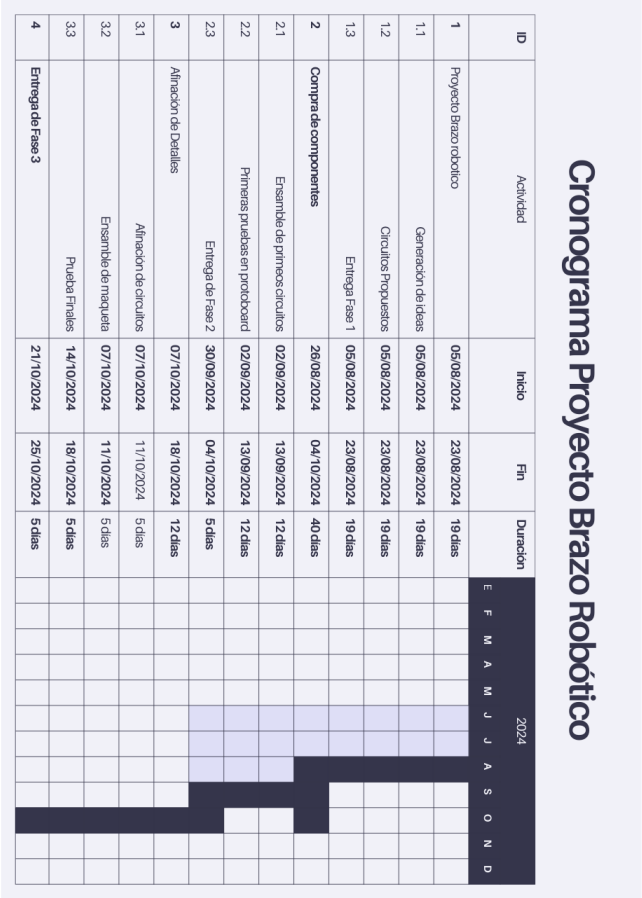
1. ¿Debemos calibrar el equipo para obtener los valores de referencia o existe alguna tabla que nos indique los valores?

2. ¿Todos los dedos tienen que funcionar al mismo tiempo?

3. ¿Qué módulo para el sensor muscular recomienda utilizar y que sea compatible con la TivaC?

4. ¿Qué precauciones debemos tener al momento de realizar el proyecto?

VI. CALENDARIZACIÓN



ID	Actividad	Inicio	Fin	Duración
1	Proyecto Brazo robotico	05/08/2024	23/08/2024	19 días
1.1	Generación de Ideas	05/08/2024	23/08/2024	19 días
1.2	Circuitos Propuestos	05/08/2024	23/08/2024	19 días
1.3	Entrega Fase 1	05/08/2024	23/08/2024	19 días
2	**Compra de componentes**	26/08/2024	04/10/2024	40 días
2.1	Ensamble de primeros circuitos	02/09/2024	13/09/2024	12 días
2.2	Primeras pruebas en protoboard	02/09/2024	13/09/2024	12 días
2.3	Entrega de Fase 2	30/09/2024	04/10/2024	5 días
3	Afinación de Detalles	07/10/2024	18/10/2024	12 días
3.1	Afinación de circuitos	07/10/2024	11/10/2024	5 días
3.2	Ensamble de maqueta	07/10/2024	11/10/2024	5 días
3.3	Prueba Finales	14/10/2024	18/10/2024	5 días
4	**Entrega de Fase3**	21/10/2024	25/10/2024	5 días

Figura 8: Calendarización

VII. FOTOGRAFÍAS DE LOS AVANCES

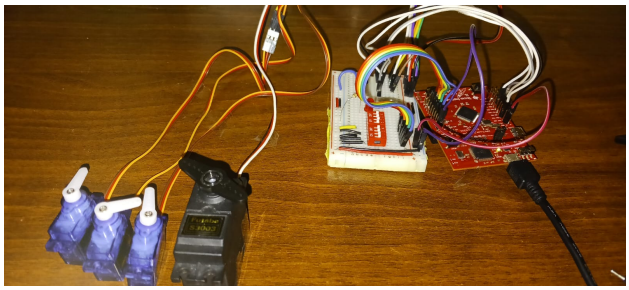


Figura 9: Vista general Tiva C TM4C123GH6PM, dipswitch y motores

Fuente: Elaboración propia, 2024

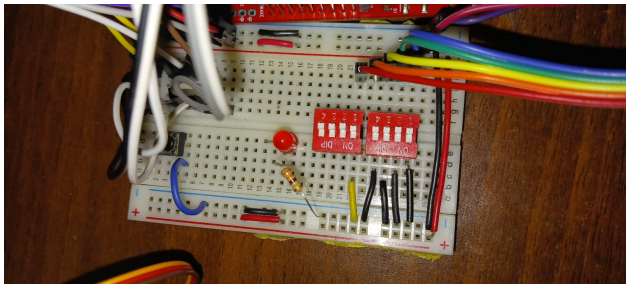


Figura 10: Vista de las conexiones GPIO con el dipswitch

Fuente: Elaboración propia, 2024

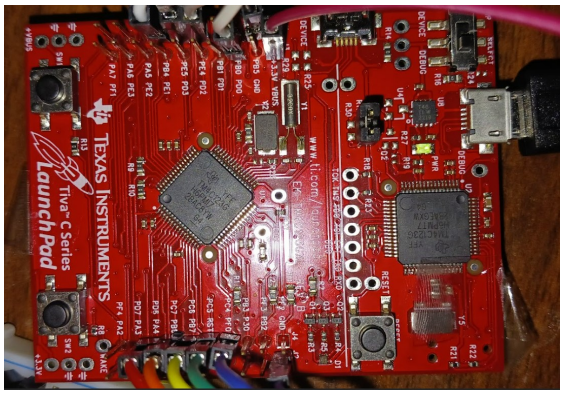


Figura 11: Vista de los pines Tiva C TM4C123GH6PM

Fuente: Elaboración propia, 2024

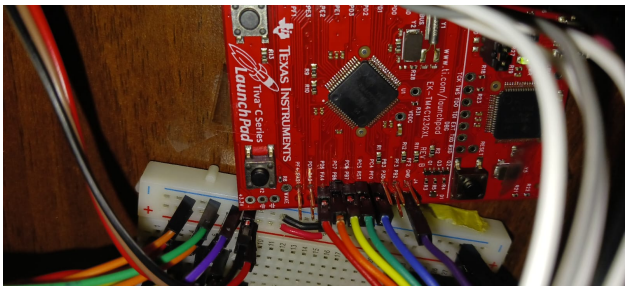


Figura 12: Vista de los pines Tiva C TM4C123GH6PM

Fuente: Elaboración propia, 2024

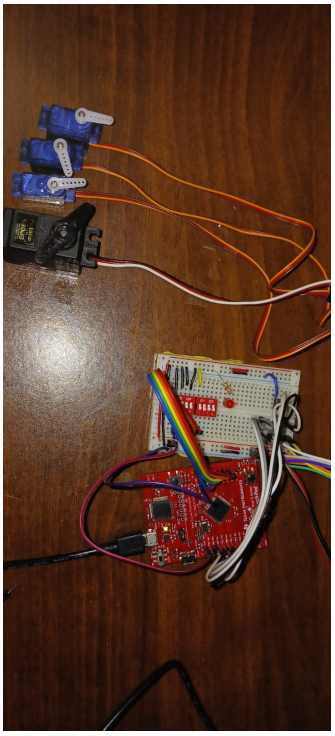


Figura 13: Vista general del arreglo de motores y conexiones

Fuente: Elaboración propia, 2024

@bookboylestad2003electronica, title=Electrónica: teoría de circuitos y dispositivos electrónicos, author=Boylestad, Robert L and Nashelsky, Louis and Barraza, Carlos Mendoza and Fernández, Agustín Suárez, volume=8, year=2003, publisher=PEARSON educación

@articlefigueroa2020control, title=Control de velocidad y sentido de giro para un motor DC., author=Figueroa, Robert Alexis Torres and Valderrama, Wilkin, journal=Infometric@-Serie Ingeniería, Básicas y Agrícolas, volume=3, number=1, pages=103–115, year=2020

@bookperez2007sistemas, title=Sistemas electrónicos digitales, author=Pérez, Enrique Mandado and Mandado, Enrique and Mandado, Yago, year=2007, publisher=Marcombo

VIII. CÓDIGO IMPLEMENTADO

```
//PC4 -- PB4
//PC5 -- PB5
//PC6 -- PE5
//PC7 -- PD0
//PD6 -- PE4

#include <stdint.h>
#include <stdbool.h>
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "inc/tm4c123gh6pm.h"
#include "driverlib/sysctl.h"
#include "driverlib/gpio.h"
#include "driverlib/pwm.h"
#include "driverlib/pin_map.h"

#define PWM_FREQUENCY 55
uint32_t Load;
uint32_t PWMClock;
uint8_t Adjust1;
uint8_t Adjust2;
uint8_t Adjust3;
uint8_t Adjust4;
uint8_t Adjust5;

//int valor_PD = 0;
//int valor_PC = 0;
//bool posicion_norte = true; // Variable para controlar la posición del servo
//int valor_PC4_anterior = 1; // Estado anterior del botón PC4

int main(void)
{

    Adjust1 = 28; // Inicializamos el valor de ajuste (posición inicial del servo)
    Adjust2 = 28;
    Adjust3 = 28;
    Adjust4 = 28;
    Adjust5 = 28;

    SysCtlClockSet(SYSCTL_OSC_MAIN | SYSCTL_USE_PLL | SYSCTL_XTAL_16MHZ |
SYSCTL_SYSDIV_2_5); // Configura el reloj a 80 MHz

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOB);  // Habilita el puerto B
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOE);  // Habilita el puerto E
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);  // Habilita el puerto D
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOC);  // Habilita el puerto C
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM0);   // Habilitar el módulo PWM0
    SysCtlPeripheralEnable(SYSCTL_PERIPH_PWM1);   // Habilitar el módulo PWM1

    SysCtlPWMClockSet(SYSCTL_PWMDIV_64); // divisor del reloj para el PWM
```

```
   GPIOPinTypePWM(GPIO_PORTB_BASE, GPIO_PIN_4 | GPIO_PIN_5); // Configurar
PB4 Y PB5 como salida
   GPIOPinTypePWM(GPIO_PORTE_BASE, GPIO_PIN_4 | GPIO_PIN_5); // Configurar
PE4 Y PE5 como salida
   GPIOPinTypePWM(GPIO_PORTD_BASE, GPIO_PIN_0); // Configurar PD0 como
salida
   GPIOPinConfigure(GPIO_PB4_M0PWM2); // Configurar PB4 M0PWM2
   GPIOPinConfigure(GPIO_PB5_M0PWM3); // Configurar PB5 M0PWM3
   GPIOPinConfigure(GPIO_PE4_M1PWM2); // Configurar PE4 M1PWM2
   GPIOPinConfigure(GPIO_PE5_M1PWM3); // Configurar PE5 M1PWM3
   GPIOPinConfigure(GPIO_PD0_M1PWM0); // Configurar PD0 M1PWM0


   //GPIOPinTypeGPIOOutput(GPIO_PORTB_BASE, GPIO_PIN_1); // Define PB1 como
salida
   //GPIOPinTypeGPIOOutput(GPIO_PORTE_BASE, GPIO_PIN_4); // Define PE4 como
salida
   GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_6);  // Define PD6 como
entrada
   GPIOPinTypeGPIOInput(GPIO_PORTC_BASE, GPIO_PIN_4 | GPIO_PIN_5 |
GPIO_PIN_6 | GPIO_PIN_7);  // Define PC4 5 6 7 como entrada
   GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_6, GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU); // Configuración pull-up en PD6
   GPIOPadConfigSet(GPIO_PORTC_BASE,GPIO_PIN_4 | GPIO_PIN_5 | GPIO_PIN_6 |
GPIO_PIN_7, GPIO_STRENGTH_2MA, GPIO_PIN_TYPE_STD_WPU); // Configuración
pull-up en PC

   // Calcular el valor de "Load" para el periodo del PWM
   PWMClock = SysCtlClockGet() / 64;
   Load = (PWMClock / PWM_FREQUENCY) - 1;

   //Configurar PB4 y PB5
   PWMGenConfigure(PWM0_BASE, PWM_GEN_1, PWM_GEN_MODE_DOWN);
//Generador 1 base pwm0 PB4 PB5
   PWMGenPeriodSet(PWM0_BASE, PWM_GEN_1, Load); // Generador 1, Base PWM0
PB4 PB5
   PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2, Adjust1 * Load / 1000); //PB4
   PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3 , Adjust2 * Load / 1000); //PB5
   PWMOutputState(PWM0_BASE, PWM_OUT_2_BIT, true); //PB4
   PWMOutputState(PWM0_BASE,PWM_OUT_3_BIT, true); //PB5
   PWMGenEnable(PWM0_BASE, PWM_GEN_1); //Generador 1, Base PWM0

   //Configurar PE4 PE5
   PWMGenConfigure(PWM1_BASE, PWM_GEN_1, PWM_GEN_MODE_DOWN);
//Generador 1 base PWM1
   PWMGenPeriodSet(PWM1_BASE, PWM_GEN_1, Load);
   PWMPulseWidthSet(PWM1_BASE, PWM_OUT_2, Adjust3 * Load / 1000); //PE4
   PWMPulseWidthSet(PWM1_BASE, PWM_OUT_3 , Adjust4 * Load / 1000); //PE5
   PWMOutputState(PWM1_BASE, PWM_OUT_2_BIT, true); //PE4
   PWMOutputState(PWM1_BASE, PWM_OUT_3_BIT, true); //PE5
   PWMGenEnable(PWM1_BASE, PWM_GEN_1); //Generador 1, Base PWM1

   PWMGenConfigure(PWM1_BASE, PWM_GEN_0, PWM_GEN_MODE_DOWN);
   PWMGenPeriodSet(PWM1_BASE, PWM_GEN_0, Load);
```

```
   PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, Adjust5 * Load / 1000);
////Establecer el ancho de pulso
   PWMOutputState(PWM1_BASE, PWM_OUT_0_BIT, true);
   PWMGenEnable(PWM1_BASE, PWM_GEN_0);


   while(1)
   {
      int valor_PC4 = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_4);
      int valor_PC5 = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_5);
      int valor_PC6 = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_6);
      int valor_PC7 = GPIOPinRead(GPIO_PORTC_BASE, GPIO_PIN_7);
      int valor_PD6 = GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_6);

      // Motor 1 (PB4) controlado por PC4
            if (valor_PC4 == 0)
            {
               Adjust1 = 140; // Mover a 180 grados
            }
            else
            {
               Adjust1 = 28; // Regresar a 90 grados
            }
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_2, Adjust1 * Load / 1000);

            // Motor 2 (PB5) controlado por PC5
            if (valor_PC5 == 0)
            {
               Adjust2 = 140;
            }
            else
            {
               Adjust2 = 28;
            }
            PWMPulseWidthSet(PWM0_BASE, PWM_OUT_3, Adjust2 * Load / 1000);

            // Motor 3 (PE5) controlado por PC6
            if (valor_PC6 == 0)
            {
               Adjust3 = 140;
            }
            else
            {
               Adjust3 = 28;
            }
            PWMPulseWidthSet(PWM1_BASE, PWM_OUT_3, Adjust3 * Load / 1000);

            // Motor 4 (PE4) controlado por PC7
            if (valor_PC7 == 0)
            {
               Adjust4 = 140;
            }
            else
            {
```

```
            Adjust4 = 28;
        }
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_2, Adjust4 * Load / 1000);

        // Motor 5 (PD0) controlado por PD6
        if (valor_PD6 == 0)
        {
            Adjust5 = 140;
        }
        else
        {
            Adjust5 = 28;
        }
        PWMPulseWidthSet(PWM1_BASE, PWM_OUT_0, Adjust5 * Load / 1000);

        SysCtlDelay(10000); // Pequeño retardo para evitar múltiples lecturas rápidas
    }

}
```