Práctica 3. Acondicionamiento de galgas.

1 Introducción

En esta práctica vamos a acondicionar unas galgas para implementar una balanza de precisión. El objetivo es diseñar un sistema para medir la deformación causada por un peso sobre una regla metálica y con ello, al ser la deformación proporcional al peso, poder medir el peso.
Para ello podemos usar las siguientes ecuaciones:

$$\sigma = \frac{e(L-d)}{2I}P, \quad \epsilon = \frac{\Delta L}{L} = \frac{\sigma}{E} \qquad E\text{: módulo de Young}$$

Equation 1 Relación entre fuerza y deformación.

Por tanto, para poder sacar la relación entre la fuerza y deformación, se van a tener que calcular una serie de variables que afectan a la relación. Afortunadamente, todas ellas salvo E (el módulo de Young), son variables geométricas, así que son fáciles de calcular.
Esta última es una variable intrínseca del material, y se tendrá que sacar mediante ensayos, ya que existe una gran variabilidad entre varas (y tanto…).

2 Circuito de acondicionamiento

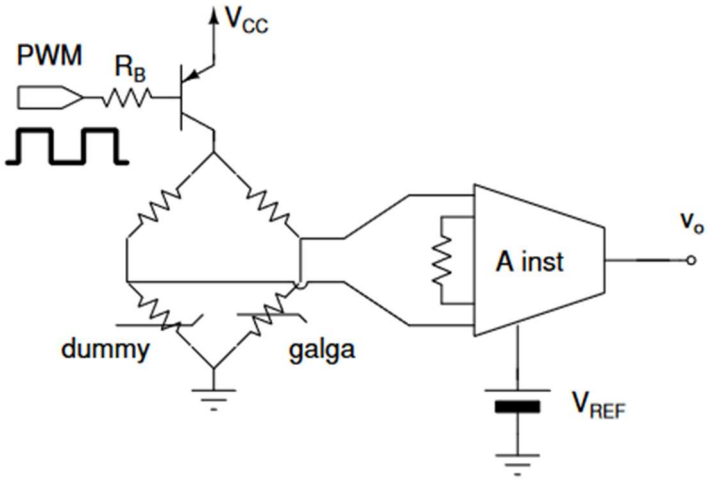


Figure 1 Implementación práctica del circuito de puente con chopping.

3 Obtención del módulo de Young

Para obtener el módulo de Young se ha realizado un ensayo para obtener la frecuencia de oscilación libre del sistema, para ello se ha colocado la salida de del circuito de acondicionamiento en un osciloscopio (con tensión de alimentación constante dentro del

puente).

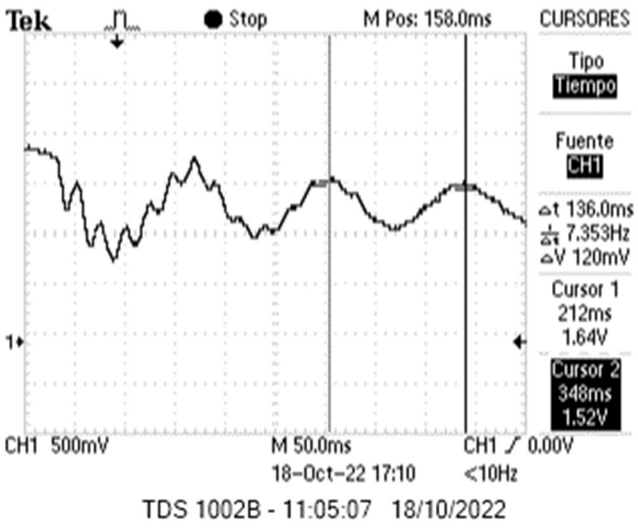


Figure 2 Respuesta del sensor ante un golpe.

Tras soltar la regla desde un punto donde está flexionada, se puede ver que esta empieza a oscilar, estas oscilaciones se producen, a grandes rasgos siguiendo una señal senoidal exponencial decreciente. Tal y como se ve en la Figure 2, la frecuencia de esta exponencial es de 7.353Hz. Esta frecuencia es por definición la frecuencia de oscilación natural.

$$\omega_p = \sqrt{\frac{EIL}{m}}\left(\frac{1{,}875}{L}\right)^2, \qquad m\text{: masa de la estructura}$$

Equation 2 Pulsación de resonancia propia de la estructura

Usando esta ecuación finalmente se ha conseguido el siguiente valor de E:

$$E = 0.74092309081 * 10^{11}\ \left[N/m^2\right] = 74[GPa]$$

Este valor es más pequeño de lo esperable, ya que el acero suele tener un valor cercano a los 210 GPa, pero está en el mismo orden de magnitud, así que se toma como correcto.

4 Medida del peso

Para medir el peso existen diferentes formas de hacerlo. En primer lugar, se puede encontrar la relación entre el peso y la tensión y usar dicha función en el microcontrolador.

En segundo lugar, se puede realizar una serie de ensayos experimentales con los que hacer una curva de calibración y calcular mediante interpolación entre medidas el peso usado.

Para esta práctica se han usado ambos métodos.

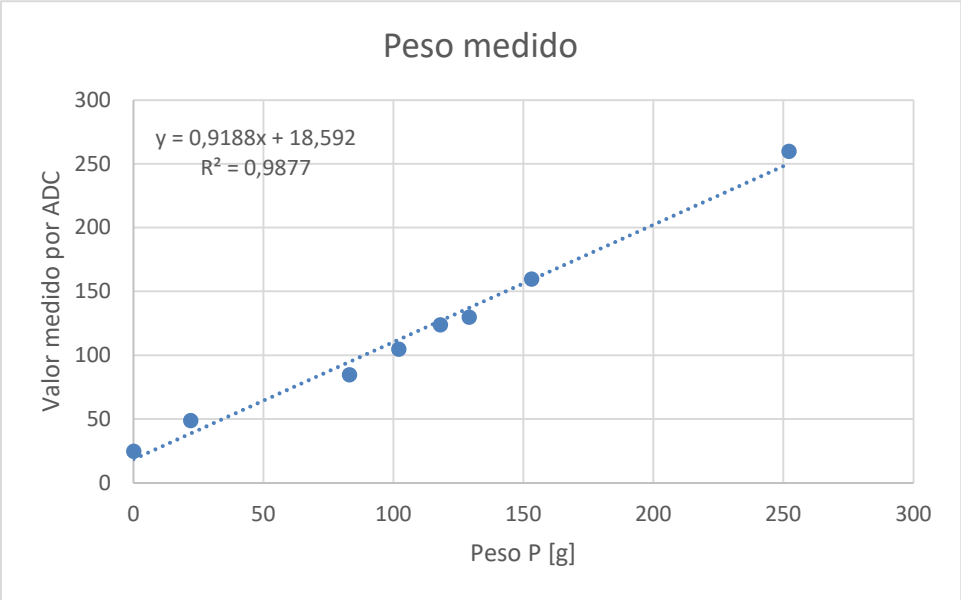
4.1 Ecuación del peso

Usando el valor del módulo de Young del apartado anterior, se puede calcular la relación entre el peso y la deformación usando la Equation 1 Relación entre fuerza y deformación.

$$P\ [N] = 6520.94 * \Delta L\ [m]$$

Por otra parte, en el Apartado2 se ha dimensionado de tal forma que se tiene la siguiente relación entre la tensión y la deformación.

4.2 Tabla de interpolación



5 Implementación balanza

Para implementar la balanza digital, es necesario alimentar el puente con una señal PWM con valor en alto de 3.3 V. Esto se hará a través de un transistor PNP conectado al micro (Figure 1) por el pin RB10. El valor de la resistencia Rb se ha calculado para que el transistor funcione siempre en modo saturación y su valor es de 820 Ω.

También se medirá el valor de la tensión de salida del amplificador de instrumentación dos veces por cada ciclo del PWM, una cuando está en alto y otra cuando está en bajo. A partir de estas dos medidas de tensión, se calcula el valor de Vo sin offset y con ella se calcula el peso por interpolación utilizando los puntos medidos del apartado 4.2.

A continuación, se presenta el código del archivo main.c. El PWM se ha configurado para que genere interrupciones en la mitad de cada semiperiodo, y la función interpolarSensor hace la interpolación para calcular el peso.

Para ralentizar el envío de datos, el bucle scan tiene un periodo de 200ms, de forma que solamente se enviarán 5 datos por segundo por la UART, que es más que suficiente para la aplicación de uso del dspic.

```
#include <p33FJ128MC802.h>
#include <xc.h>
#include <stdio.h>
#include <stdlib.h>
#include "uart.h"
#include "config.h"
#include "pwm.h"
#include "idle.h"
#include "adc.h"
#include "interrupciones.h"
#include "interpolar_sensor.h"

void sendData(void);

float buffer[2][2];

int bandera = 0;

int main(void)
{
    char send_data[9];
    inicializarReloj();
    TRISB &= 0x0FFF;
    // Inicializaciones
    inicializarUART(115200);
    inicializarTareaIdle(2000);// 200 ms
    inicializarADCPolling(1<<5);
    inicializarPWM((1 << 10), 1000); //1000 Hz y se puede hasta 6kHz
    setDcPWM(1<<10, 5000);
    activarPWM((1 << 10)); //entrada puente pin RB10



    IEC3bits.PWM1IE = 1; //habilitar interrupción
    IPC14bits.PWM1IP = 3; //prioridad interrupción pwm
    IFS3bits.PWM1IF = 0; //flag pwm

    Enable(); //se habilitan las interrupciones


    while (1)
    {
        if(bandera == 1){
            sendData();
            bandera = 0;

        }

        tareaIdle();


    }
        return 0;
}

void sendData(void)
{
    unsigned int e1, e2, s1, s2;
    unsigned int peso = 0;
    unsigned int tension = 0;
```

```
    Disable();
    e1 = buffer [0][0];
    e2 = buffer [1][0];
    s1 = buffer [0][1];
    s2 = buffer [1][1];
    Enable();

    if (e1 == 0 && e2 == 1) {

        tension = s1 - s2;

    }
    else {

        tension = s2 - s1;

    }

    peso =  InterpolarSensor(tension);

    char send_data[25];
    sprintf(send_data, "peso = %d", peso);
    putsUART(send_data);
}


void __attribute__((interrupt(no_auto_psv))) _MPWM1Interrupt (void) {

    static int posbuffer = 0; //posición de guardado
    static int cont = 0;
    unsigned int salida = 0;
    unsigned int entrada = 0;

    IFS3bits.PWM1IF = 0;

    salida = leerADCPolling(5); //conectar salida a AN5

    if(P1TMRbits.PTDIR == 1){
        entrada = 0;
    }else{ entrada = 1;}

    cont++;


    buffer[posbuffer][0] = entrada;
    buffer[posbuffer][1] = salida;

    posbuffer =! posbuffer;


    if(cont == 2){
        bandera = 1;
        cont = 0;
    }

}
```