**Práctica 2 b. Control de intensidad luminosa**

En la segunda parte de esta práctica se ha programado el a tarjeta PicTrainer V3 para realizar la conversión A/D de la medida del sensor, y obtener mediante interpolación lineal el valor de luminosidad. Este valor se mandará por UART al ordenador para poder monitorizarlo.

# Conversión A/D

En primer lugar, se conectará la salida del circuito de acondicionamiento del sensor a una de las

entradas analógicas de la tarjeta. En este caso el AN5, es decir, el pin RB3. Por tanto, tenemos que inicializar el ADC para ese pin y, en este caso al estar usando un bucle scan, incluir la función, (tarea) que lea los datos del ADC. Este devuelve un valor entre 0 y 1023 en el buffer ADC1BUF0, por tanto usamos un unsigned int para guardar los datos.

int main(void) {

…

    inicializarADCPolling(0x20);

    unsigned int lectura;

    while (1) {

…

        lectura = leerADCPolling(5);

        …

    }

    return 0;

}

# Obtención de la luminosidad mediante la curva de calibración

Para obtener el valor de la medida en lúmenes tenemos que hacer la conversión entre el numero en la escala de 10bit a el valor de luminosidad requerido. Anteriormente, en la primera parte de la práctica hemos calculado la relación entre la tensión en AN5 y los lúmenes. En la documentación del fabricante (DSPIC33 ADC 16.15) se puede ver que la relación entre la tensión y los valores generados por el ADC es lineal sin ningún offset, por lo que nos basta con hacer un cambio de escala.

## Interpolación lineal

Para calcular el valor visto por el ADC se va a usar el método de interpolación lineal, ya que resulta el más rápido, especialmente si la relación entre entrada y salida no se puede expresar fácilmente con una función polinómica de orden bajo.

Para ello se guardan una serie de valores de la entrada y salida de tal forma que se aproximará la luminosidad mediante los dos valores contiguos en la tabla. Para encontrar dichos valores, se entrará en la tabla buscado entre que dos índices del vector está el valor del ADC y se entrará con estos mismos en la función.

#include <xc.h>

#include "interpolar\_sensor.h"

unsigned int interpolarSensor ( unsigned int valor\_adc ){

    int pos = 0;

    unsigned int lux;

    static unsigned int vec\_adc [15] = {0,  124,   133,  174,  217,   248,   353,  412,  527,   620,   682,   713,   821,  992, 1023};

    static unsigned int vec\_lux [15] = {0, 75, 58,120 , 102, 154, 186, 320, 380,  540,  640, 700, 820, 1100, 1279};

    while(valor\_adc > vec\_adc[pos + 1]){pos++;} //valor\_adc estar� entre pos y pos+1

    lux = vec\_lux[pos]+(((vec\_lux[pos+1] - vec\_lux[pos])/(vec\_adc[pos+1] - vec\_adc[pos]))\*(valor\_adc - vec\_adc[pos]));

    if(lux<=0)lux=0; // no hace falta, pero para filtrar posibles errores

    return lux;

}

El valor máximo y mínimo han de ser los extremos del rango de salida del ADC en este caso al ser de 10 bits, 0/1023.

Esta función, se llamará en el main desde el bucle scan para traducir la medida y luego se guardará en la variable lux para usarse en otras tareas.

int main(void) {

    unsigned int lux;

    while (1) {

…

        lux = interpolarSensor(lectura);

…

    }

    return 0;

}

# Comunicación con el PC

El valor de la luminosidad obtenido se mandará al PC, para eso se usará otra tarea llamada EnviarUART, que tendrá como argumento el dato a mandar.

void EnviarUART ( unsigned int lux) {

static char send\_data[12];

    sprintf(send\_data, "lux: %u\n", lux);

    putsUART(send\_data);

}

El dato que se manda por la UART, para hacer más fácil su entendimiento se manda en el formato "lux: %u\n", para en caso de tener varios mensajes con funciones diferentes, saber que representa cada uno. Para ello lo que hacemos es hacer una cadena de caracteres de longitud 12 donde ponemos el texto. Con la palabra lux, el espacio y el salto de línea consumimos 6 caracteres, y dado que la luminosidad llega hasta valores de 4 cifras, debemos tener un mínimo de 10 caracteres en la cadena.

Para ejecutar esta tarea, llamamos a la función en el bucle scan. Pero es importante recordar inicializar el UART antes.

int main(void) {

    …

    inicializarUART(115200);

…

    while (1) {

…

EnviarUART(lux);

…

    }

    return 0;

}

# Periodo del bucle scan.

Por último, nos piden que el periodo del bucle scan sea de 100ms para ello, deberíamos de medir el tiempo que tarda cada tarea y poner un retraso extra para llegar al valor deseado. Pero en este caso se ha decidido despreciar el tiempo de las tareas.

Por tanto, el código sería el siguiente:

int main(void) {

    …

    inicializarTareaIdle(1000)

    …

    while (1) {

        tareaIdle();

        …

    }

    return 0;

}