Raspberry Pi - Libraries

22.07 - Programación I

1) DISPLAY
2) JOYSTICK
3) AUDIO

DISPLAY (MATRIZ DE LEDS)

- La librería que usaremos para la matriz de LEDs se llama disdrv. Ustedes cuentan con el código objeto (disdrv.o) y con el *header* (disdrv.h).
- La matriz de LEDs es de 16 x 16. Es importante tener en cuenta que el origen (0,0) está el extremo superior izquierdo.
- En el *header* están definidas algunas constantes de interés, como los valores máximo y mínimo de las coordenadas x e y.
- En disdrv.h pueden ver que la librería proporciona cuatro funciones.

void disp_init(void);

- Sirve para inicializar el display.
- Debe ser llamada únicamente una vez antes de empezar a usarlo.
- Ejemplo:

```
disp_init();
```

void disp_clear(void);

- Permite, al mismo tiempo, borrar todo el contenido del *buffer* y apagar todos los LEDS
- Nota: en el *buffer* se van almacenando los cambios hechos con disp_write().
- Ejemplo:

```
disp_clear();
```

void disp_write(dcoord_t coord, dlevel_t val);

- Prende o apaga (según val) el LED indicado en coord.
- dcoord_t es una estructura que contiene las coordenadas x e y.
- dlevel_t tiene dos valores posibles: D_OFF o D_ON, para apagado o prendido.
- disp_write() escribe en el *buffer*. Por lo tanto, las llamadas a disp_write() no se ven automáticamente en el display. Para eso debemos usar disp_update().
- Ejemplo:

```
dcoord_t myPoint = {1,5};

disp_write(myPoint, D_ON);
```

void disp_update(void);

- Actualiza el display con el contenido del *buffer*.
- Notemos que con múltiples llamadas a disp_write() podemos modificar muchos LEDS y luego mostrar el resultado con una llamada a disp_update().
- Ejemplo:

```
disp_update();
```

Ejemplo: testDisp

- testDisp enciende todo el display, de a un LED, y por filas.
- Usen este ejemplo para probar que el display funciona correctamente.

```c
/*
testDisp enciende todo el display de a un LED por vez
*/

#include <stdio.h>
#include "disdrv.h"
#include "termlib.h"

int main(void)
{

    dcoord_t myPoint = {};      //inicializa myPoint en (0,0). Recordemos que está arriba a la izquierda.
    disp_init();                //inicializa el display
    disp_clear();               //limpia todo el display
    for (myPoint.y = DISP_MIN; myPoint.y <= (DISP_MAX_Y); myPoint.y++)  //para cada coordenada en y...
    {
        for (myPoint.x = DISP_MIN; myPoint.x <= (DISP_MAX_X); myPoint.x++)  //para cada coordenada en x...
        {
            disp_write(myPoint, D_ON);              //prende el LED en el buffer.
            printf(YELLOW_TEXT "(%2d,%2d) ", myPoint.x, myPoint.y);//imprime cuál LED fue encendido
            disp_update();                          //ahora lo vemos en el display
        }
        printf("\n");
    }
    return 0;
}
```

JOYSTICK

- La librería para el *joystick* es joydrv. Ustedes cuentan con el código objeto (joydrv.o) y con el *header* (joydrv.h).
- El *joystick* se puede mover en el plano *XY*. Las coordenadas toman valores entre JOY_MAX_NEG (-127) y JOY_MAX_POS (128), constantes útiles y definidas en el *header*.
- Además, tiene un interruptor (*switch*), que se activa al presionarlo.
- En joydrv.h pueden ver que la librería proporciona cuatro funciones.

void joy_init(void);

- Sirve para inicializar el *joystick*.
- Por lo tanto, debe llamarse únicamente una vez antes de empezar a usarlo.
- Ejemplo:

```
joy_init();
```

void joy_update(void);

- Cuando se llama **se miden** las coordenadas del *joystick* **en ese momento**.
- Luego podemos obtener esas coordenadas usando joy_get_coord() o joy_get_switch().
- Ejemplo:

```
joy_update();
```

jcoord_t joy_get_coord(void);

- Devuelve las coordenadas del *joystick* medidas en la última llamada a joy_update().
- La estructura jcoord_t contiene las coordenadas x e y para el *joystick*. No confundir con el tipo de datos dcoord_t que se usa para las coordenadas en el display.
- Ejemplo:

```
jcoord_t myCoords;

myCoords = joy_get_coord();
```

jswitch_t joy_get_switch(void);

- Devuelve el estado del *switch* medido en la última llamada a joy_update().
- jswitch_t tiene dos valores posibles: J_NOPRESS o J_PRESS.
- Ejemplo:

```
jswitch_t mySwitch;

mySwitch = joy_get_switch();
```

Ejemplo: testJoy

- testJoy imprime las coordenadas del joystick. Termina cuando se presiona el switch.
- Usen este ejemplo para probar que el joystick funciona correctamente.

```c
/*
testJoy imprime las coordenadas del joystick. Termina cuando se presiona el switch.
*/

#include <stdio.h>
#include "joydrv.h"
#include "termlib.h"

int main(void)
{
    jcoord_t myCoords;
    joy_init();             //inicializo el joystick
    do
    {
        joy_update();                           //primero actualizo las coordenadas medidas
        myCoords = joy_get_coord();             //luego las guardo en myCoords
        printf(CYAN_TEXT "(%4d, %4d)\n", myCoords.x, myCoords.y);
    } while (joy_get_switch() == J_NOPRESS);    //mientras no se presione el switch

    return 0;
}
```

AUDIO

- La librería que usamos para el audio se llama libaudio. Ustedes cuentan con el código objeto (libaudio.o) y el header (libaudio.h).
- Permite reproducir audio de forma no bloqueante. Los archivos deben ser .wav.
- En libaudio.h pueden ver que la librería proporciona seis funciones.

int init_sound(void);

- Inicializa la librería. Por lo tanto, debe llamarse una sola vez al principio.
- Ejemplo:

```
init_sound();
```

int set_file_to_play(const char *music_path);

- Abre el archivo a reproducir indicado por music_path (no lo reproduce).
- Recordemos que el formato de audio debe ser .wav.
- Ejemplo:

```
char mySong[] = "song.wav";
set_file_to_play(mySong);
```

Nota formato de audio: WAV (Microsoft) signed 16 bit PCM

int play_sound(void);

- Inicia la reproducción del archivo de audio. Como ya dijimos, no es bloqueante.
- Ejemplo:

```
play_sound();
```

int pause_sound(void);

- Pausa la reproducción del archivo de audio.
- Ejemplo:

```
pause_sound();
```

int stop_sound(void);

- Se detiene la reproducción del archivo de audio y se cierra el mismo.
- Se usa para terminar la reproducción.
- Ejemplo:

```
stop_sound();
```

int player_status(void);

- Devuelve el estado del sistema. Hay seis estados posibles:
  - **NO_INIT**: sistema no inicializado, es decir, no se invocó init_sound().
  - **READY**: sistema inicializado, listo para usar.
  - **STOPPED**: archivo de audio listo para ser reproducido.
  - **PLAYING**: reproduciendo audio.
  - **PAUSED**: reproducción pausada.
  - **FINISHED**: el archivo terminó de ser reproducido.

Ejemplo: testAudio

- testAudio reproduce el soundtrack de Mario Bros por un minuto.
- Usen este ejemplo para probar que el audio funciona correctamente.

```c
#include <stdio.h>
#include <pthread.h>
#include <stdlib.h>
#include "libaudio.h"
#include <SDL/SDL.h>

#define SONG_NAME "./Sound/mariobros.wav" //nombre del archivo de audio
int main(void)
{
    char mySong[] = SONG_NAME;
    printf("Audio player \n");

    init_sound();                       //inicializa el driver
    if (player_status() == READY)       //si la inicialización fue existosa sigo
    {
        set_file_to_play(mySong);       //carga el archivo a reproducir
        play_sound();                   //reproduce el archivo de audio (no bloqueante)
        printf("Playing: %s \n", mySong);
    }
    else
    {
        printf("System not ready. Did you run init_sound() ? \n");
        return 1;
    }
    while (player_status() == PLAYING) {}   //espera que el archivo termine de reproducirse
    printf("Finished: %s\n", mySong);
    return 0;
}
```

Extra: control de volumen

- El volumen del audio se puede modificar mediante consola, entre 0% y 90%.
- Ejemplo:

```
amixer sset PCM,0 90%
```