

Sistema de iluminación de leds

Integrantes del grupo

Lis Muñoz Fortea: lis.munoz.forteaa@alumnos.upm.es

Teresa Ramírez Haro: teresa.ramirez.haro@alumnos.upm.es

Silvia Saeta Álvarez: s.saetaaa@alumnos.upm.es

Resumen del trabajo

El objetivo del proyecto es crear un sistema de iluminación de leds que funcione tras de verificar un código introducido por un teclado.

Disponemos de un programa en C que pedirá al usuario que elija el led que quiere encender y que estará funcionando hasta que se seleccione la opción de salir. Este programa pedirá un nuevo número si se introduce uno que no está entre las opciones que propone el programa.

Tras enviar el número del led el programa de arduino esperará hasta que el sensor de ultrasonidos detecte un objeto a una distancia menor que 10 centímetros. Cuando esto ocurra, se encenderá la pantalla LCD que solicitará una clave definido en el código de arduino.

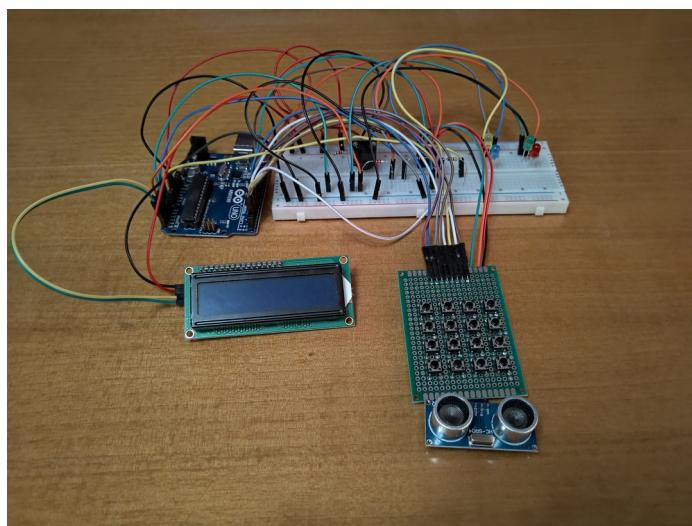
La clave se introduce a través de un teclado matricial, en el que las teclas que se pulsen irán escribiéndose en la pantalla. El teclado tiene la opción de borrar un solo carácter pulsando la tecla '*' y de borrar todos los caracteres introducidos hasta el momento con la tecla '#' para evitar errores durante la introducción del código a través del teclado matricial.

Si se introduce una clave incorrecta se solicitará otra y además sonará un pitido a través del buzzer. Por el contrario, si se ha introducido el código correcto hay dos opciones, en las dos sonará un pitido indicando que el código es correcto. La primera es que el led seleccionado en el ordenador no estuviera encendido, en este caso, se apagaría el led que sí lo estuviera, en el caso de que hubiera alguno encendido, y se encenderá el seleccionado. Por otra parte, si hemos elegido un led que ya estaba encendido, tras introducir la clave correcta, este se apagará.

También contiene medidas de seguridad, por ejemplo, se apaga la pantalla si han pasado 10 segundos sin escribir el primer carácter para solucionar posibles fallos si el sensor de ultrasonidos detecta algo que no debería. Otra medida de seguridad es que se apaga la pantalla si se han introducido 5 contraseñas incorrectas. Además, el ultrasonidos no detectará nada y por tanto no se encenderá la pantalla hasta que se hayan enviado los datos desde el ordenador.

Cuando se selecciona la opción de salir en el ordenador, el programa de C termina y el arduino dejará de funcionar, al no estar activo el ultrasonidos, aunque sí se mantendrán encendidos los leds que lo estuvieran antes de salir.

Hardware



Para la creación del proyecto hemos utilizado: placa arduino UNO, protoboard, sensor de ultrasonidos, pantalla lcd i2c 16x2, teclado matricial 4x4, leds, resistencias para los leds y un buzzer.

-Sensor de ultrasonidos

El sensor de ultrasonidos tiene dos pines, Trigger y Echo. Desde el Trigger se envía una señal, que posteriormente recibirá el Echo tras encontrarse la señal con un objeto y rebotar. Midiendo el tiempo que tarda el Echo en recibir la señal desde que el Trigger la envía se puede calcular la distancia a la que se encuentra un objeto.

Para utilizarlo hemos usado la siguiente función:

```
int medir_distancias (int TriggerPin, int EchoPin)
{
    long tiempo, distancia_cm;
    digitalWrite (TriggerPin, LOW);
    delayMicroseconds (4);
    digitalWrite (TriggerPin, HIGH);
    delayMicroseconds (10); //El Trigger envía una señal durante 10 microsegundos.
    digitalWrite (TriggerPin, LOW);
```

```

        tiempo = pulseIn (EchoPin HIGH); //Mide el tiempo que tarda el EchoPin en pasar
                                         de LOW a HIGH.
        distancia_cm = (tiempo * 1 / 29.2)/ 2; //Se calcula la distancia a partir del tiempo y la
                                         velocidad del sonido y se pasa a
                                         centímetros.
    return (distancia_cm);
}

```



-Leds

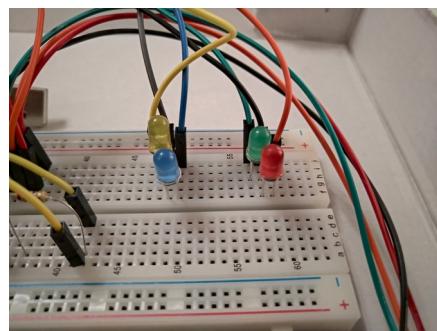
Para los leds solo han sido necesarias dos funciones que los enciendan y los apaguen:

```

void encender (int LedPin)
{
    digitalWrite (LedPin, HIGH);
}

void apagar (int LedPin)
{
    digitalWrite (LedPin, LOW);
}

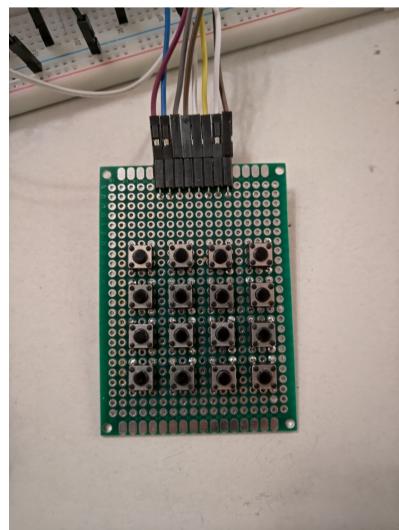
```



-Teclado matricial

Un teclado matricial consiste en la agrupación de botones individuales colocados en filas y columnas formando una matriz 4x4. Este teclado tiene como ventaja frente a los botones individuales que el número de pines necesarios es menor.

Para manejar el teclado hemos utilizado la librería Keypad.h con funciones como el getKey(), que guarda la tecla del teclado pulsada.



-Pantalla lcd i2c

La pantalla lcd i2c es una pantalla lcd con un adaptador que nos permite usarla con sólo dos pines, lo que es una gran ventaja frente a la lcd normal, con la que necesitamos siete. Sus pines están configurados por defecto en el A4(SDA) y A5 (SCL). Por ello al declararla solo necesitamos poner lcd(0x3F,16,2), los pines se configuran por defecto según la dirección i2c, que en este caso es 0x3F.

Para manejarla usaremos la librería LiquidCrystal_I2C.h, con funciones como lcd.init, que la configura e inicializa, lcd.clear, para borrar la pantalla, lcd.backlight, para encenderla, lcd.setCursor, para colocar el cursor en las coordenadas que queramos, o lcd.print para escribir en la pantalla.

La pantalla utilizada en este caso es de 16x2.



-Buzzer

El buzzer es un altavoz que convierte una señal eléctrica en una onda de sonido. Se utiliza con la función tone, que recibe su pin, la frecuencia del sonido emitido y su duración. En nuestro caso, hemos emitido sonidos diferentes en función de si la contraseña introducida es la correcta o no.



Otras funciones utilizadas

Se necesitará una función para elegir cuál es el led que se enciende según el dato que le llegue del puerto serie:

```
int elegir_pin (char c)
{
    int resultado;

    switch (c)
    {
        case '0':resultado=LedPinAmarillo; break;
        case '1':resultado=LedPinAzul; break;
        case '2':resultado=LedPinVerde; break;
        case '3':resultado=LedPinRojo; break;
    }
    return (resultado);
}
```

Utilizaremos una función para escribir en la pantalla lo que se haya introducido por teclado:

-Función para escribir en la pantalla:

La función tendrá como entrada una cadena de caracteres en la que se habrían guardado las teclas presionadas en el teclado por el usuario.

```

void escribir_en_lcd (char caracter, int *posicion_x, int *posicion_y)
{
    int i;

    for (i=0;i<N;i++)
    {
        lcd.setCursor (*posicion_x, *posicion_y);
        lcd.print (caracter);
        (*posicion_x)++;
    }
}

```

Además, usaremos una función para comparar el código correcto con el introducido por el usuario. Esta función devolverá 1 o 0 en función de si la clave es correcta o no:

```

int comprobacion_clave (char cadena[ ], char codigo[ ])
{
    int resultado=1, i;

    for (i=0;i<N;i++)
    {
        if (cadena[i]!=codigo[i])
            resultado=0;
    }
    return (resultado);
}

```

Funciones para contar el tiempo:

Será necesaria la librería time.h. Para estas funciones también usaremos funciones de la librería time.h como now() que guarda el tiempo en ese momento.

Usaremos una estructura tiempo:

```

typedef struct
{
    float tiempo_inicial, tiempo_final;
    float intervalo_de_tiempo;
}tiempo;

```

La primera función será para inicializar el tiempo inicial y el final a 0:

```

tiempo inicializar_tiempo (tiempo t)
{
    t.tiempo_inicial=now();
    t.tiempo_final=now();
}

```

```

t.intervalo_de_tiempo=(float) (t.tiempo_final-t.tiempo_inicial);
return t;
}

```

La segunda función guarda el tiempo actual en `tiempo_final` y calcula el intervalo de tiempo:

```

tiempo tiempo_transcurrido (tiempo t)
{
t.tiempo_final=now();
t.intervalo_de_tiempo= (float) (t.tiempo_final-t.tiempo_inicial);
return t;
}

```

Funciones para borrar caracteres escritos:

La primera de ellas será para borrar un solo carácter. Se le pasará la cadena de caracteres, la posición en la que se encuentra el carácter para borrar en ese momento y la posición de las x en la pantalla lcd:

```

char borrar_caracter (char cadena[], int *n, int *pos_x)
{
char aux;
cadena[(*n)-1]= ' '; //Se cambia el carácter por un espacio en blanco.
aux=cadena[(*n)-1];
(*n)--; //Se disminuye en 1 la posición en la que estamos guardando los caracteres en la cadena.
(*pos_x)=(*pos_x)-1; //Se disminuye la posición en la que escribimos en el lcd.
return aux; //Devolvemos un valor auxiliar que corresponde al carácter y no el carácter porque al haber disminuido n cadena[(*n)-1] no es la que ha cambiado.
}

```

La siguiente función es para borrar todos los caracteres de la cadena. Se le pasa la posición en la que está la cadena y la posición de las x en el lcd:

```

void borrar_todo (char cadena[], int *n, int *pos_x)
{
int i, posicion_borrado_x=0;
for (i=0;i<(*n);i++) //Se cambian todos los caracteres por espacios y se escriben en la pantalla.
{
    cadena[i]=' ';
    escribir_en_lcd (cadena[i], &posicion_borrado_x, &posicion_y);
}
(*n)=0; //Se cambia la posición en la cadena y la posición en las x en el lcd a 0.
(*pos_x)=0;
}

```

Comunicación entre C y arduino

-En el programa de C:

Creamos un puntero a un tipo FILE que abriremos usando fopen("/dev/ttyACM0","w"). Esto abre el fichero del puerto serie del arduino en modo escritura para enviar datos. Tras esto, según el número elegido se envía un valor diferente al arduino. Para esto usamos una función que contiene las siguientes instrucciones:

```
fprintf(nombre,"%d",valor); //Se envía el numero de salida del led  
sleep(3);  
fflush(nombre);
```

El fflush se utiliza para que se envíe el número que hemos escrito.

No abrimos y cerramos el fichero cada vez que queramos enviar datos porque al abrir el puerto serie el arduino se resetea y no se guardan los datos y por tanto, los leds no se mantendrían encendidos al enviar el siguiente dato.

Tras pulsar la tecla de salir en el programa cerramos el fichero con fclose.

-En el programa de arduino:

Esperamos a que reciba un valor a partir del puerto serie. El programa no continuará hasta que le llegue algún valor

```
while (input=='\0')  
{  
    if(Serial.available()>0)  
    {  
        input=Serial.read(); //Guarda el valor enviado en la variable input  
    }  
}  
LedPin = elegir_pin (input); //Asigna un valor a LedPin dependiendo del que ha llegado del  
puerto serie
```

Bibliografía

Para el hardware:

-Led:

<https://www.luisllamas.es/encender-un-led-con-arduino/>

-Sensor de ultrasonidos:

<https://programarfacil.com/tutoriales/fragmentos/sensor-de-ultrasonidos-con-arduino/>

<https://www.luisllamas.es/medir-distancia-con-arduino-y-sensor-de-ultrasonidos-hc-sr04/>

-Pantalla LCD:

<https://www.luisllamas.es/arduino-lcd-hitachi-hd44780/>

<https://www.geekfactory.mx/tutoriales/tutoriales-pic/pantalla-lcd-16x2-con-pic-libreria/>

https://www.naylampmechatronics.com/blog/35_Tutorial--LCD-con-I2C-controla-un-LCD-con-so.html

<https://www.arduino.cc/en/Reference/LiquidCrystal>

-Teclado matricial:

<https://www.luisllamas.es/arduino-teclado-matricial/>

<http://arduparatodos.blogspot.com/2017/12/teclado-matricial-con-arduino-varios.html>

-Buzzer:

<https://www.luisllamas.es/reproducir-sonidos-arduino-buzzer-pasivo-altavoz/>

Para la comunicación entre C y arduino:

<https://salilkapur.wordpress.com/2013/03/08/communicating-with-arduino-using-c/>

<https://www.luisllamas.es/enviar-recibir-numeros-puerto-serie-arduino/>

<https://stackoverflow.com/questions/30291943/communicating-between-c-arduino>

<https://forum.arduino.cc/index.php?topic=306198.0>

<https://linux.die.net/man/3/sleep>