Sistemas Embebidos - Memoria práctica 2.

Elvi Mihai Sabau Sabau[1][51254875L]

[1] Universidad de Alicante, Alicante, España.
emss5@alu.ua.es

Abstract. En esta práctica usaremos un arduino nano 33 IoT con headers para realizar 2 ejercicios, uno relacionado con el uso de la bombilla LED para familiarizarnos con el estilo de programación, y otro realización con el módulo WiFi del dispositivo, conectándolo a una red, y mostrando mensajes de error vía serial y también usando de la bombilla LED.

Keywords: IoT, Arduino, LED, WiFiNINA.

1 Practica 2.

1.1 Descripción.

Esta práctica está partida en 2 ejercicios, el primero que es el de desarrollar un programa capaz de hacer que la bombilla LED parpadee en 5 secuencias, del 1 al 5, ej: ●, ●●, ●●●, ●●●●, ●●●●●. y el segundo el de realizar un programa capaz de hacer que el dispositivo se conecte de forma automática a una red WiFi, y mediante el bus de serie (bus que usaremos para comunicar mensajes) y el led muestre su estado, siendo la secuencia del ejercicio anterior la secuencia que se reproducirá si la conexión ha fallado.

Además, aunque el enunciado no lo ha pedido, he decidido añadirle la función de reconectar automáticamente, y de mostrar dichos estados por el bus de mensajes.

1.2 Programa.

1.2.1 Función led_blink.

A continuación explicaremos la función “led_blink”, que reproduce la secuencia de parpadeos del 1 al 5.

```
void led_blink(int del_time, int seq_time) {
    for (int i = 1; i <= 5; i++) {
        for (int j = 1; j <= i; j++ ) {
            digitalWrite(LED_BUILTIN, HIGH);
            delay(del_time);
            digitalWrite(LED_BUILTIN, LOW);
            delay(del_time);
        }

        // Time delay between blink sequences.
        delay(seq_time);
    }
}
```

Código. 1 Código de la función led_blink.

Para realizar dicha secuencia de parpadeos, hemos usando 2 bucle, uno dentro de otro, el bucle interno para el parpadeo, y el externo para la secuencia del 1 al 5.

La función recibe por parámetro el tiempo en más de espera entre parpadeos, y el tiempo de espera entre secuencias.

El enunciado especifica 2 cosas sobre esta función, que el tiempo de espera entre parpadeos debe ser 100 ms, y que esta función se ejecutará cuando el dispositivo falle al conectarse a la red especificada, también que muestre un mensaje de error si tras 5 seg. no consigue conectarse a la red.

Para ello, al llamar a esta función podemos especificar los tiempos 100ms para la espera entre parpadeos, y 700 entre secuencias, de esta manera conseguimos alargar la animación a 5 segundos ((5+4+3+2+1) * 100 + 5 * 700 = 5000ms).

Como he mencionado anteriormente, además de parpadear, el dispositivo intentará reconectarse a la red nuevamente.

1.2.2 Funciones para conectarse a una red..

Para realizar este ejercicio, deberemos importar la librería WiFiNINA, esto se puede hacer desde el gestor de librerías de Arduino IDE.

Fig. 1 Captura del gestor de librerías de Arduino IDE.

Una vez instalado, podremos importarlo y tendremos acceso a las funciones necesarias para interactuar con el módulo WiFi del dispositivo.

En nuestro caso usaremos 2 funciones de esta librería y una constante, una para conectarnos a la red, y otra para comprobar el estado de conexión usando la constante.

```
WiFi.begin(SSID, PASSWORD);

if (WiFi.status() != WL_CONNECTED) Serial.print("Not connected!");
if (WiFi.status() == WL_CONNECTED) Serial.print("Connected!")
```

Código. 2 Ejemplo de uso de la librería WiFiNINA para conectarse a una red, y para comprobar el estado de la conexión..

1.2.3 Programa completo.

Los programas en arduino suelen tener 2 funciones principales, “setup” y “loop” setup es una función que se ejecutará solo la primera vez, al arrancar el dispositivo, y “loop” se ejecutará en bucles mientras el dispositivo siga encendido.

En nuestro setup, nos encargamos de configurar inicialmente el dispositivo, configurando el nivel de baudios para la transmisión de mensajes vía bus serie, y de intentar inicialmente conectarnos a la red WiFi y dejar el LED encendido.

Además, tendremos unas variables globales: el SSID de la red, la contraseña, y una variable que usaremos para detectar cuando el dispositivo está conectado y cuando no, de esta manera evitaremos que el dispositivo ejecute en bucle ciertas tareas que queremos que se ejecute solo una vez una vez el estado de su conexión haya cambiado.

Crearemos además otras funciones independientes para controlar el voltaje de los LEDs, y para conectarnos a la red.

De esta manera, en nuestra función “loop” comprobaremos inicialmente la conexión, si está conectado, apagamos el LED y mostramos un mensaje de notificación al usuario.

En caso contrario, apagamos el LED, iniciamos la animación de “led_blink”, mostramos el mensaje de error tras 5 segundos, y intentamos conectar.

En caso de que previamente el dispositivo estuviera conectado, y en la siguiente iteración haya cambiado a estado desconectado, también lo notificaremos al usuario.

El código de esta práctica está adjunto al fichero comprimido donde se aloja este mismo documento.

1.3 Ejecución.

A continuación:

1) Arrancaremos el dispositivo sin que la red esté operativa forzando la muestra del mensaje de error.

2) Encendemos la red, el dispositivo se conectará automáticamente, el dispositivo mostrará el mensaje de conexión.

3) Entonces apagaremos la red, y al desconectarse mostrará nuevamente el mensaje de error.

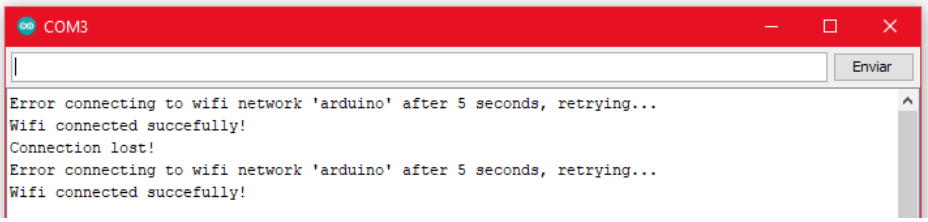

Fig. 2 Captura monitor serie de Arduino IDE, mostrando los mensajes emitidos por el dispositivo.

2 Conclusiones.

En esta práctica hemos aprendido lo básico para maniobrar un dispositivo arduino nano 33, la manera de programar correctamente en las funciones “setup” y “loop” y la manera de interactuar con el led y el módulo WiFi del dispositivo.