

Universidad de San Carlos de Guatemala
Primer semestre
Facultad de ingeniería
Escuela de Ciencias y Sistemas
Arquitectura de Computadoras y Ensambladores 1



Roberto Carlos Gómez Donis 202000544

Edgardo Andrés Nil Guzmán 20180119

César André Ramírez Dávila 202010816

José Manuel Ibarra Pirir 202001800

Angel Francisco Sique Santos 202012039

Materiales

- Pantalla LCD de 16x4
- Botones
- Teclado alfanumérico matricial de 3x4
- Módulo Bluetooth
- Arduinos
- Matriz de LEDs 8x8
- Driver para matriz de LEDs MAX7219
- Sensor de temperatura LM35
- Resistencias que considere necesarias

Definición de variables importantes para el funcionamiento del sistema.

```
1 #define LOOP while (true)
2
3 // MENUS
4 const int SECUENCIA_INICIAL = 0, MENU_PRINCIPAL = 1, LOGIN = 2, REGISTER = 3;
5 ADMIN = 4, CLIENTE = 5, ESCOGER_TECLADO = 6, ADMIN_LOGS = 7, ADMIN_STATUS = 8;
6 INGRESAR_CELULAR = 9, RETIRAR_CELULAR = 10, USER_DELETE = 11;
7
8 int menu_actual = SECUENCIA_INICIAL;
9
10 // PINES
11 const int rs = 12, en = 11, d4 = 5, d5 = 4, d6 = 3, d7 = 2;
12 LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
13
14 const int din = 10, cs = 9, clk = 8;
15 LedControl matriz(din, clk, cs, 1);
16
17 // Variable auxiliar Nombre usuario
18 String auxNombre = "";
19 int intentos_fallidos = 0;
20 int intentos_fallidos_login = 0;
21 int intentos_fallidos_login_globales = 0;
22 int celulares_ingresados = 0;
23 int usuarios_activos = 0;
24 int cantidad_incidentes = 0;
25 // bool button_.state = false;
26
27 int contador_logs = 0;
```

Definición de botones que serán importantes en el proyecto.

```
29 // BOTONES
30 Button Btn_Ok(23);
31 Button Btn_Cancel(24);
32 Button Btn_Comp1(45);
33 Button Btn_Comp2(46);
34 Button Btn_Comp3(47);
35 Button Btn_Comp4(48);
36 Button Btn_Comp5(49);
37 Button Btn_Comp6(50);
38 Button Btn_Comp7(51);
39 Button Btn_Comp8(52);
40 Button Btn_Comp9(53);
```

Funciones para el programa.

```
41 /* Usuario */
42 Usuarios authenticated_user;
43 Log log_generado;
44
45 int letra_actual_index = 0;
46 // FUNCIONES
47 void menu_setup();
48 void menu_loop();
49 void conectar_dispositivo();
50 void limpiarBuffer();
51
52 // *MENUS
53 void secuencia_inicial();
54 void menu_principal();
55 void login();
56 void registro();
57 void admin_logs();
58 void admin_status();
59 void loop_status();
60 void loop_logs();
61 void letras_matriz();
62 void menu_administrar();
63 void menu_cliente();
64 void menu_seleccionar_teclado(int menu_ingresos);
```

```
54 //
55 void ingreso_celular();
56 void retiro_celular();
57 void retirar_dispositivo(Cajas compartimento);
58
59 void eliminar_cuenta();
60 void simulate_button_state();
61 /// FUNCIONES
```

```
63 void menu_setup() {  
64   lcd.begin(16, 4); // Inicializa una pantalla LCD con 16 columnas y 4 filas.  
65  
66   matriz.shutdown(0, false); // Desactiva el modo de apagado de una matriz de LED en el índice 0.  
67   matriz.setIntensity(0, 8); // Configura la intensidad de la matriz de LED en el índice 0 a 8.  
68   matriz.clearDisplay(0);    // Borra la pantalla de la matriz de LED en el índice 0.  
69  
70   Btn_Ok.setup();           // Configura un botón llamado "ok_button".  
71   Btn_Cancel.setup();       // Configura un botón llamado "cancel_button".  
72 }
```

Este menú es donde se estará ejecutando la aplicación

```

74 void menu_loop() {
75     simulate_button_state();
76     switch (menu_actual) {
77         case SECUENCIA_INICIAL:
78             secuencia_inicial();
79             break;
80
81         case MENU_PRINCIPAL:
82             menu_principal();
83             break;
84
85         case ESCOGER_TECLADO:
86
87             menu_seleccionar_teclado(menu_ingresos);
88             break;
89
90         case LOGIN:
91             login();
92             break;
93
94         case REGISTER:
95             registro();
96             break;
97
98         case ADMIN:
99             menu_administrar();
100             break;
101
102         case CLIENTE:
103             menu_cliente();
104             break;
105
106         case ADMIN_LOGS:
107             admin_logs();
108             break;
109
110         case ADMIN_STATUS:
111             admin_status();
112             break;
113
114         case INGRESAR_CELULAR:
115             ingreso_celular();
116             break;
117
118         case RETIRAR_CELULAR:
119             retiro_celular();
120             break;
121         case USER_DELETE:
122             eliminar_cuenta();
123             break;
124
125         default:
126             break;
127     }

```

Esta función es usada con la aplicación bluetooth para mandar una cadena.

```
128 void enviarConfirmar(char *cadena) {  
129     Serial1.println(cadena);  
130     bool hayAlgo = false;  
131     char recibidos[3];  
132     LOOP {  
133         while (Serial1.available()) {  
134             Serial1.readBytes(recibidos, 2);  
135             hayAlgo = true;  
136         }  
137         if (hayAlgo && !Serial1.available())  
138             break;  
139     }  
140 }  
141  
142 }
```

```

144 void secuencia_inicial() {
145     lcd.clear(); // Borra la pantalla LCD.
146
147     lcd.setCursor(0, 0); // Establece el cursor en la posición (1, 0) de la pantalla LCD.
148     lcd.print("Bienvenido"); // Imprime el mensaje "Bienvenido" en la pantalla LCD.
149
150     // Imprime diferentes nombres en la pantalla LCD.
151     delay(300); // Pausa de 0.3 segundos
152     lcd.setCursor(0, 1); // Establece el cursor en la posición (0, 1) de la pantalla LCD.
153     lcd.print("Roberto202000544");
154
155     delay(300); // Pausa de 0.3 segundos
156     lcd.setCursor(0, 2);
157     lcd.print("Edgardo201801119");
158
159     delay(300); // Pausa de 0.9 segundos
160     lcd.setCursor(0, 3);
161     lcd.print("Cesar 202010816");
162
163     delay(300); // Pausa de 0.3 segundos
164     lcd.setCursor(0, 4);
165     lcd.print("Jose 202001800");
166
167     delay(300); // Pausa de 0.3 segundos
168     lcd.setCursor(0, 5);
169     lcd.print("Angel 202012039");
170
171     while (true) {
172         if (Btn_Ok.is_pressed()) {
173             menu_actual = MENU_PRINCIPAL;
174             break;
175         }
176     }
177 }

```


La función llamada "recibir_texto_app" toma dos parámetros: un puntero a una matriz de caracteres llamada "mensaje" y un puntero a una matriz de caracteres llamada "titulo". La función devuelve una cadena. El propósito de la función es mostrar un mensaje con lo requerido en la aplicación móvil y pedirle al usuario que ingrese lo solicitado para recibirlo en el arduino.

```
String recibir_texto_app(char *mensaje, char *titulo) {
    bool termino = false;
    LOOP {

        //Serial.println("Entro a loop nombre");
        if (termino) {
            break;
        }
        termino = true;
        limpiarBuffer();
        enviarConfirmar(mensaje);
        memset(nombre_temp, 0, 11);
        lcd.clear();
        lcd.print(titulo);
        lcd.setCursor(0, 1);
        lcd.print(" - ");
        lcd.print(mensaje);
        lcd.print(":");
        lcd.println("");
        lcd.setCursor(0, 2);
        // OBTENER CADENA DE APLICACIÓN -- Nombre
        bool seEnvioAlgo = false;
        int indiceNombre = 0;
        long int t0 = millis();
        long int t1 = millis();
        limpiarBuffer();

        LOOP {
            // SI YA SE ENVIO ALGO DESDE LA APLICACION
            while (Serial1.available()) {
                seEnvioAlgo = true;
                // RECIBIRLO
                nombre_temp[indiceNombre++] = Serial1.read();

                Serial.println(nombre_temp);
                //Serial.println("Entro a en enviado algo");
            }
            // CONTROLAR CUANTO HA PASADO DESDE QUE COME...
            if (seEnvioAlgo) {
                t1 = millis();
                if (t1 - t0 >= 500)
                    break;
            } else {
                t0 = millis();
                t1 = millis();
            }
        }
    }
    Serial.print("DATOS OBTENIDOS DESDE APP");
    return nombre_temp;
}
```

La función llamada "conectar_dispositivo" se utiliza para establecer una conexión con un dispositivo. Primero borra la pantalla LCD y muestra un mensaje "Esperando una conexión..." . Luego espera a que los datos estén disponibles en el puerto Serial1 y lee datos en la matriz "recibidos". Una vez que se reciben los datos, establece el indicador "alguienPorAhi" en verdadero y sale del ciclo. Luego borra la pantalla LCD y muestra un mensaje "Conectado con éxito".

```
void conectar_dispositivo() {
    limpiarBuffer();
    lcd.clear();
    lcd.print(" Esperando una ");
    lcd.setCursor(0, 1);
    lcd.print("  conexion...  ");
    bool alguienPorAhi = false;
    char recibidos[3];
    LOOP {
        while (Serial1.available()) {
            Serial1.readBytes(recibidos, 2);
            alguienPorAhi = true;
        }
        if (alguienPorAhi && !Serial1.available())
            break;
    }
    limpiarBuffer();
    lcd.clear();
    lcd.print("Conectado con éxito");
    delay(1000);
    lcd.clear();
    Serial.println("CONECTADO CON EXITO");
}
```

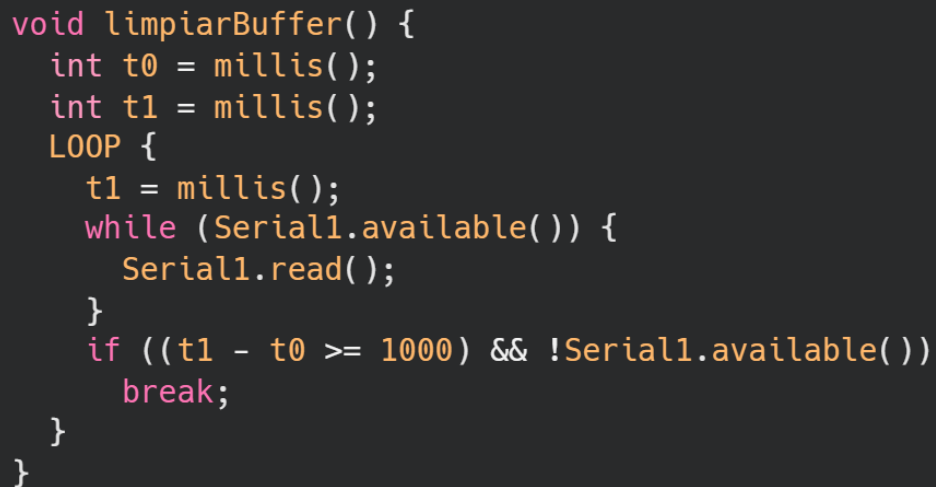
La función llamada "enviar_texto_app" toma como parámetros un mensaje de cadena y un contador de enteros. La función envía un mensaje a un dispositivo utilizando el protocolo de comunicación Serial1 y espera una respuesta. En esta función se envía la palabra mostrar para recibirla en la app y que identifique que el mensaje solo se debe mostrar y no debe pedir nada como en "recibir_texto_app".

```
void enviar_texto_app(String mensaje, int contador) {
    Serial1.println("Mostrar ," + mensaje + String(contador));
    bool hayAlgo = false;
    char recibidos[3];

    while (true) {
        while (Serial1.available()) {
            Serial1.readBytes(recibidos, 2);
            hayAlgo = true;
            //Serial.println("LOOP 2 ENVIAR TEXTO");
        }

        if (hayAlgo && !Serial1.available()) break;
        //Serial.println("LOOP 1 ENVIAR TEXTO");
    }
    delay(3000);
}
```

El código anterior define una función llamada "limpiarBuffer" que borra el búfer de entrada del puerto de comunicación Serial1. Lo hace leyendo y descartando continuamente cualquier dato disponible en el búfer hasta que no queden más datos o hasta que haya pasado un segundo. Esta función se puede utilizar para garantizar que el búfer de entrada esté vacío antes de recibir nuevos datos.



```
void limpiarBuffer() {  
    int t0 = millis();  
    int t1 = millis();  
    LOOP {  
        t1 = millis();  
        while (Serial1.available()) {  
            Serial1.read();  
        }  
        if ((t1 - t0 >= 1000) && !Serial1.available())  
            break;  
    }  
}
```