Restaurante Inteligente

Santiago Cadena, Mateo Huertas, Tomas Cuartas
Departamento de Ingeniería Eléctrica y Electrónica, Universidad Nacional de Colombia
Bogotá, Colombia
scadenaa@unal.edu.co
shuertast@unal.edu.co
tcuartas@unal.edu.co

***Abstract*— En el siguiente documento se presentará la elaboración de un restaurante inteligente, el cual incluye un mesero inteligente (pantalla LCD), que interactúa con el usuario recibiendo sus órdenes mediante un teclado, así como también mostrando el estado de su pedido y emitiendo un sonido (buzzer) cuando está listo; el pedido será entregado al usuario mediante una banda transportadora.**

I. INTRODUCCIÓN

El restaurante inteligente es un restaurante automatizado con autoservicio, el cual entrega las comidas que se pidieron inicialmente. El programa tiene una rutina en la cual hay interacción directa con el usuario.

El desarrollo de la tecnología nos ha permitido automatizar muchos procesos y al mismo tiempo optimizarlos para que funcionen de la mejor manera; no se estaría cuerdo si no se utilizara esta herramienta para mejorar el servicio al cliente en los restaurantes, mediante la automatización de tareas tan laboriosas como las de los meseros, que no solo les quitaría una carga de encima y ayudaría a que se enfocarán en menos tareas, sino que generaría un mejor servicio y rendimiento en el restaurante.

Si se implementa a nivel macro, ya sea para otra utilidad (no necesariamente un restaurante), representa una gran ventaja, ya que no se necesitan empleados o meseros y por ende, se reducirán costos.

II. MARCO TEÓRICO

A. Placa Arduino *UNO*

Es una placa conformada principalmente por un oscilador y microcontrolador, el cual se programa con un lenguaje de programación basado en C + +. Esta placa permite realizar de manera sencilla tareas automatizadas electrónicamente.

B. Pantalla LCD

Es una pantalla de cristal líquido compuesta por píxeles que están delante de una fuente de luz y funciona gracias a la orientación de moléculas de cristal respecto al campo eléctrico con filtros y superficie reflectante. El formato escogido en el proyecto es de 2 filas y 16 columnas. Sus pines son:

- Enable:habilitador para escribir o leer una operación de la lcd.
- Register Select o RS: selector para enviar un comando o información.
- Read/Write o RW :selector para leer o escribir de la lcd), pines de datos.
  - 8 Pines de datos: Pines a los cuales se le envía la información.
  - Pines de Alimentación y contraste: Estos pines tienen la finalidad de ajustar el contraste de la pantalla LCD. Vdd es la alimentación (no debe exceder 6V), V0 es el pin del contraste y Vss es la tierra de referencia.

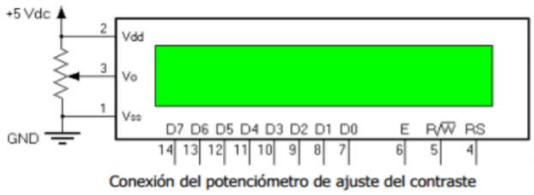


Figura 1. Conexión para ajuste de contraste en pantalla LCD.

C. Bocina

Es un dispositivo transductor que convierte la energía eléctrica en sonido. Hay bocinas o buzzers tanto activos(tienen un oscilador electrónico incorporado que funciona aplicando una tensión DC) como pasivos (funcionan de forma piezoeléctrica sin oscilador, aplicando una señal a cierta frecuencia).

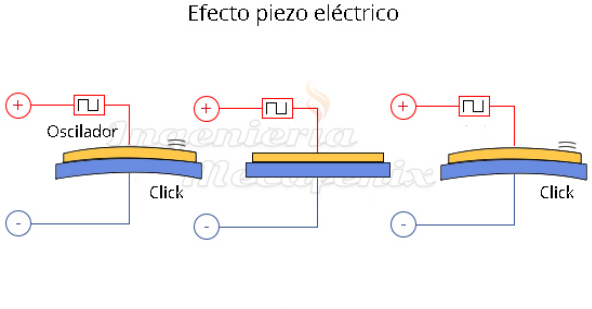


Figura 2. Efecto piezoeléctrico.

D. Teclado Matricial

Para el proyecto se hizo uso de un teclado 4x4, el cual es un arreglo de botones ordenados por filas y columnas. Para la detección de cualquier pulsación se hace un barrido de filas o columnas a 5V muy rápido, conociendo el pin que tenga una corriente entrante, se sabrá cual tecla se pulso.

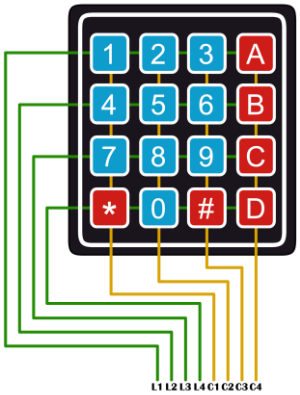


Figura 3. Teclado Matricial.

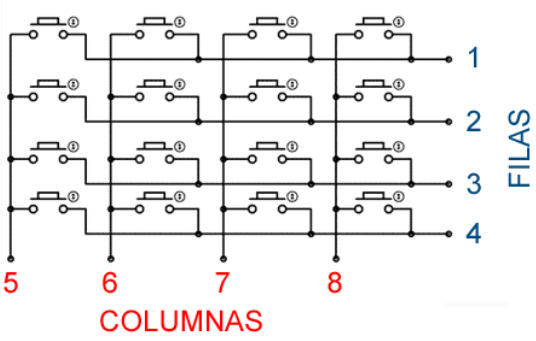


Figura 4. Filas y Columnas del teclado.

E. Motorreductor

Es un motor DC que está junto a una caja multiplicadora, la cual tiene la función de elevar la velocidad angular del eje del rotor. Con una señal en una de sus entradas señal PWM (modulación por ancho de pulso) se puede obtener una velocidad variable en el eje del rotor.

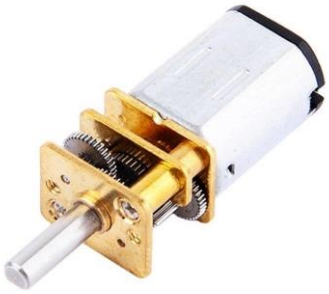


Figura 4. Motorreductor.

III. DESARROLLO DE CONTENIDOS

A. Montaje y circuito esquemático

Se utilizó una maqueta para modelar el prototipo, la cual contiene la banda transportadora.

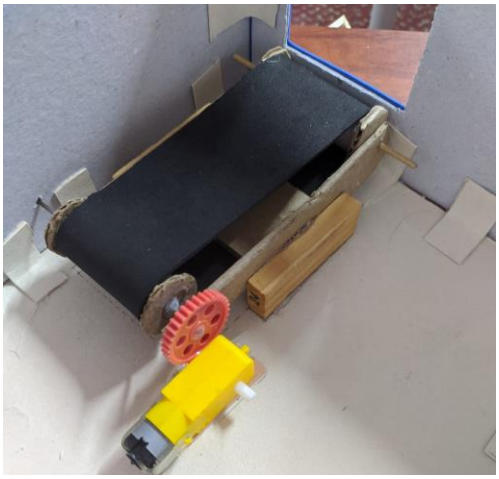
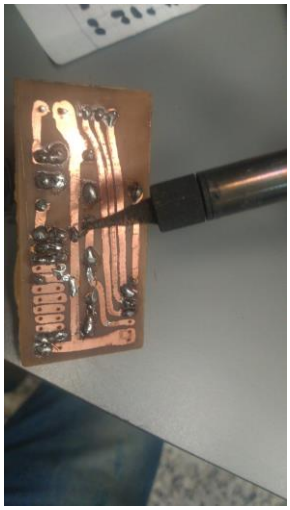


Figura 5. Maqueta realizada.



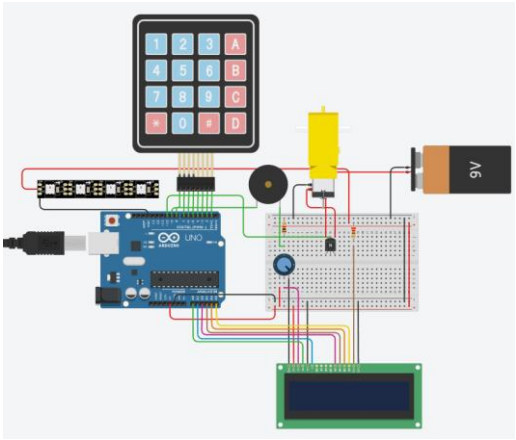


Figura 6. Montaje para el prototipo "Restaurante inteligente".

En cuanto a los aspectos más relevantes del montaje, está el uso de un transistor NPN 2N222 para aumentar la corriente de control que llega al motor y el ajuste del brillo de la pantalla

LCD con un potenciómetro.

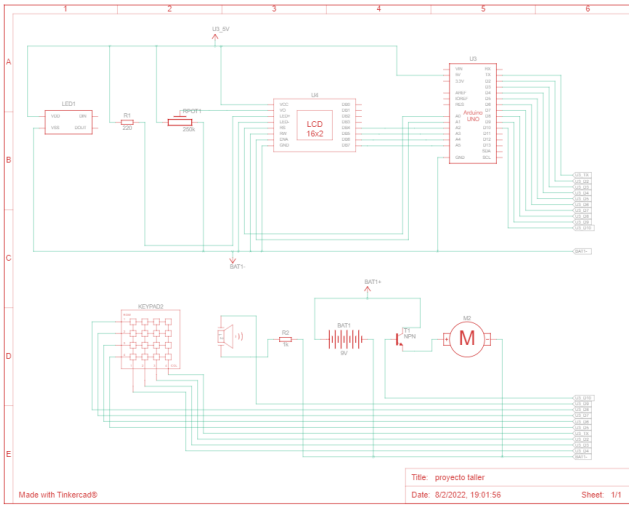


Figura 7.Circuito esquemático para el prototipo "Restaurante inteligente".

B. Código implementado

El código que se creó está en el repositorio establecido en *github[1]*.

Primeramente, se importan las librerías *LiquidCrystal.h y Keypad.h*, las cuales servirán para manejar la pantalla LCD y el teclado matricial respectivamente.
Con los pines asignados, se crea una instancia de cada clase importada y así se inicializa la pantalla lcd junto con el teclado.

```
const byte rows = 4;
const byte cols = 4;
char keyMap [rows] [cols] = { //define the cymbols on the buttons of the keypad

  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
//pins of the keypad
byte rowPins [rows] = {8, 7, 6, 5};
byte colPins [cols] = {4, 3, 2, 1};
//inicializacion de instancia myKeypad
Keypad myKeypad = Keypad( makeKeymap(keyMap), rowPins, colPins, rows, cols);
//inicializacion de instancia LiquidCrystal
LiquidCrystal lcd (A0, A1, A2, A3, A4, A5); // pins of the LCD. (RS, E, D4, D5, D6, D7)
```

Figura 8.Inicialización para la pantalla LCD y el teclado

Posteriormente, en la función *loop()* se procede a realizar la estructura del programa con una sentencia switch que va a recibir como parámetro la variable *state*. El switch va a consistir de 3 casos:

- caso 0: Tendrá un mensaje de bienvenida con el efecto de movimiento en la pantalla.
- caso 1: Saldrá el menú con las opciones del tipo de comida que se desea: hamburguesa, perro caliente, pizza o salchipapa.
- caso 2: Es la entrega del pedido, en el cual se desactiva una alarma si y sólo si el usuario teclea un dígito. Si sucede esto, la banda transportadora se activará.

```
switch(state){ //maquina de estados del programa
  case 0:  //bienvenida

    if(myKeypad.getKey()){
       state=1;
       updateMenu(menu);
       break;
     }

     lcd.setCursor(0,0);
     lcd.print(" Haz tu pedido!");
     lcd.setCursor(0, 1);
     lcd.print(Scroll_LCD_Left("Bienvenido/a al restaurante S&T&M"));
     delay(100);
     break;

  case 1: //menu
  {
       char option=myKeypad.getKey();
       if(option=='C'){ //down
            menu++;
            updateMenu(menu);
            delay(100);
            break;
            }
       else if(option=='A'){ //up
            menu--;
            updateMenu(menu);
            delay(100);
          break;
       }
       else if(option>='1'){ // digitos validos: '1' al '9'
            lcd.write(option);
            cantidad[menu-1]=option;
            delay(100);
      }

       else if(option=='*'){
            alerttime=starttime; //inicia el conteo
            asignar_turno(memory,cantidad); //asigna turno y cantidades a la mem ram
            lcd.clear();
            lcd.setCursor(0,0);
            lcd.print("Pedido Enviado");
            lcd.setCursor(0,1);
            lcd.print("Turno ");
            lcd.print(turno_entrada);
            delay(3000);
            //cantidad[] tiene los valores por default
            for(int i=0;i<4;i++){
              cantidad[i]="";}
            state=0;
            break;
       }
    break;
  }

  ,
  case 2: //entrega de pedido
    alerttime=9000000;
  if(thisNote<notes * 2){
        divider = melody[thisNote + 1];
        if (divider > 0) {// regular note
          noteDuration = (wholenote) / divider;
        }
        else if (divider < 0) {// dotted notes
          noteDuration = (wholenote) / abs(divider);
          noteDuration *= 1.5; // increases the duration in half for dotted notes
        }
        tone(buzzer, melody[thisNote], noteDuration * 0.9);
        delay(noteDuration);
        noTone(buzzer);
        thisNote=thisNote+2;
  }
  else{
    state=0;
  }
```

Figura 9. Switch con los estados del programa

En la siguiente imagen se muestra a detalle el diagrama de estados que se siguió para la lógica del programa.

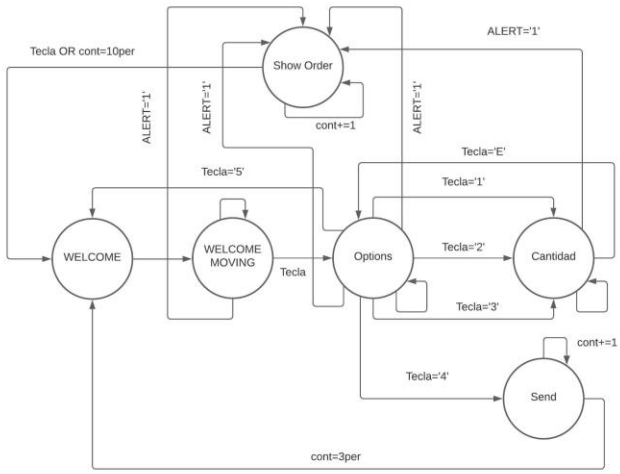


Figura 10. Diagrama de estados del programa

A parte de la función principal, también se utilizaron otras 3 funciones más.

Scroll_LCD_Left(String StrDisplay) es la función para realizar el efecto de mover las letras en el display. Se hizo de esta forma y no con un método de la instancia de la librería incorporada, ya que se tienen que realizar otras cuestiones en Arduino al tiempo, por lo que no se utiliza la programación secuencial sino orientada a eventos.

```
String Scroll_LCD_Left(String StrDisplay){
  String result;
  String StrProcess = "                " + StrDisplay + "
  result = StrProcess.substring(Lii,Li);
  Li++;
  Lii++;
  if (Li>StrProcess.length()){
    Li=16;
    Lii=0;
  }
  return result;
}
```

Figura 11.Scroll_LCD_Left(String StrDisplay)

updateMenu(int menu) actualiza la pantalla lcd, según la opción que desee el usuario con el teclado (⬆⬇).

```
void updateMenu(int menu) {
  String str_menu="      MENU      ";
  lcd.clear();
  lcd.print(str_menu);
  lcd.setCursor(0, 1);
  switch (menu) {
    case 0:
      menu = 1;
      break;
    case 1:{
      lcd.print("Hamburguesas:");
      lcd.print(cantidad[0]);
      break;}
    case 2:
      lcd.print("Salchipapas:");
      lcd.print(cantidad[1]);
      break;
    case 3:
      lcd.print("Perro Caliente:");
      lcd.print(cantidad[2]);
      break;
    case 4:
      lcd.print("Pizza:");
      lcd.print(cantidad[3]);
      break;
    case 5:
      menu = 4;
      break;
  }
```

Figura 12.updateMenu(int menu)

asignar_turno(String(&myarray)[10][4],String cantidad[4]) es la función que se utiliza para asignar el turno correspondiente cuando se envía el pedido con base en los turnos anteriores y la disponibilidad de clientes. Se pasa el arreglo de arreglos creada inicialmente (memoria RAM) por referencia, el cual es de tamaño 10, y cada elemento a su vez tiene 4 elementos (cantidades de los platos solicitado)s.

```
void asignar_turno(String(&myarray)[10][4],String cantidad[4])
{
  turno_entrada++;
  for (int i=0;i<4;i++){
    myarray [turno_salida][i]=cantidad[i];}
}
```

IV.CONCLUSIONES

- Este restaurante inteligente le da a los clientes una mejor experiencia y servicio.
- Mediante la automatización de la recepción, emisión y entrega de pedidos se puede optimizar la comunicación cliente-cocina y reducir la carga laboral de los trabajadores.
- Se puede disminuir la interacción y contacto físico entre el personal y el cliente, mitigando la propagación de enfermedades como el Covid.
- El coste para este prototipo es bajo ya que únicamente se utiliza un microcontrolador y 5 periféricos relativamente económicos.
- A nivel macro las señales emitidas por arduino harían parte de un circuito de control, el cual se utilizaría en un circuito con unos

valores nominales más altos tensión y corriente para el motor, la pantalla LCD, el teclado y demás periféricos.

- La sintaxis usando Arduino es bastante sencilla, lo que costó tiempo y trabajo fue realizar la estructura del programa para realizar distintas tareas al mismo tiempo, es decir una programación orientada a eventos no secuenciales.

REFERENCIAS

[1]santiagokdena,Restaurante-Inteligente[Online]Available:

https://github.com/santiagokdena/Restaurante-Inteligente

[2]LCD display datasheet. [Online]Available:

https://www.tme.eu/Document/213d55418e745f85348874722eeb4e3c/4DLCD-32320240.pdf

[3]DC Motor datasheet [Online] Available:

http://www.sytrans.es/docs/cms/documento-1-1349106651.pdf

[4]Passive Buzzer datasheet [Online] Available:

https://www.mouser.com/datasheet/2/400/ef532_ps-13444.pdf