

Facultad de Ingeniería
Escuela Ciencias y Sistemas
Arquitectura de Computadores y
Ensambladores 1
Sección "A"



USAC
TRICENTENARIA
Universidad de San Carlos de Guatemala

Proyecto 1

Manual Técnico

Grupo #5

201709450	Marvin Daniel Rodriguez Felix
201344708	Jose Manuel Lara Elias
201404341	Ocsael Neftali Ramirez Castillo

INTRODUCCIÓN

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar.

El presente documento presenta un proyecto en el que mediante una aplicación Android conectada al sistema por medio de Bluetooth y el microcontrolador Arduino brinda servicios de parqueo y automatiza los procesos como creación de usuarios, manejos de token, manejo de espacios disponibles de parqueo.

DESCRIPCIÓN DEL PROBLEMA

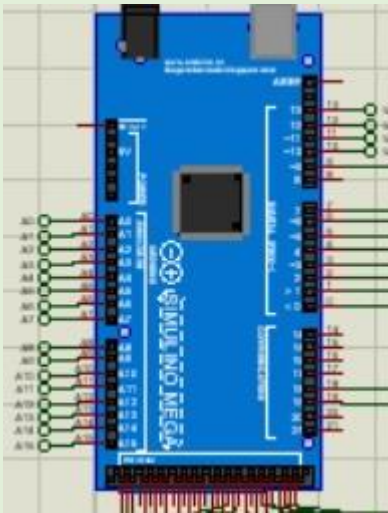
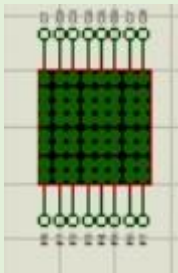
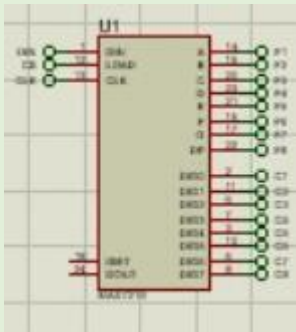
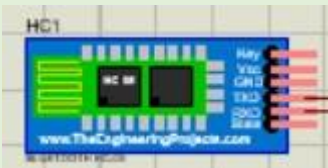
Al iniciar la simulación en la pantalla LCD se debe mostrar un mensaje de bienvenida al sistema por 10 segundos, luego de esto el sistema debe esperar que una aplicación Android se conecte por medio de Bluetooth. Al solicitar acceso al parqueo, el sistema debe de generar un token de 4 dígitos y 2 letras, el cual debe ser enviado a la aplicación del celular, luego de esto el usuario deberá ingresar este token a través de un teclado matricial de 4x4. El token ingresado debe coincidir con el token que se generó dentro del sistema , esta opción puede cancelarse lo cual hará sonar una alarma y también si este token se ingresa incorrectamente 3 veces. Si la persona que necesita usar el sistema no se encuentra registrada, se debe realizar el debido registro desde la aplicación móvil, cada usuario deberá tener máximo 8 caracteres y la contraseña 5. Estos dos campos deben guardarse en la memoria EEPROM de Arduino. Al momento de que un usuario ingrese sus datos para iniciar sesión en el sistema se deberá de ir a esta memoria y leer los datos correspondientes para verificar que el usuario existe y mostrar un mensaje en la LCD ya se que exista o no exista.

Cuando el usuario ingrese correctamente su token, se simulara que se abre la puerta mediante el control de un motor stepper, que funcionara por 10 segundos, al terminar de abrir la puerta, se tendrá una espera de 5 segundos para simular el ingreso de un vehículo, luego de esto se deberá activar otro motor, que funcionara también por 10 segundos, lo cual funcionara para simular el cierre. Se deberá de contar con 16 espacios de parqueo los cuales deben distinguirse de alguna forma ya sea en la app como en el sistema simulado, si no existe parqueo disponible se debe mostrar un mensaje en la LCD y en la app. Cada espacio en el parqueo deberá de simular una interrupción de luz con una fotorresistencia para saber si este se encuentra ocupado. Se deberá guardar en la EEPROM los parqueos para no perder la información de los reservados, esto por si el sistema pierde la energía en algún momento. Al momento que el usuario quiera salir del sistema, esto se hará a través de un botón en la aplicación, y se simulará exactamente el mismo flujo de cuando el usuario entró, el primer motor se moverá durante 10 segundos, luego se tendrá una pausa de 5

En la aplicación móvil se manejan dos tipos de usuarios, el usuario regular y el usuario administrador en este usuario podrán listarse todos los usuarios registrados. En la app se muestran dos opciones: Iniciar sesión y registrarse. Al iniciar sesión el usuario debe ingresar sus datos y el token que le fue asignado el sistema entonces mostrara la opción de si desea ingresar en ese momento o solo reservar el parqueo.

3

EQUIPO UTILIZADO

Equipo	Imagen
<u>Arduino Mega</u>	
<u>Matriz LED 8x8</u>	
Driver para matriz LED 8x8 (MAX7219)	
Modulo Bluetooth	

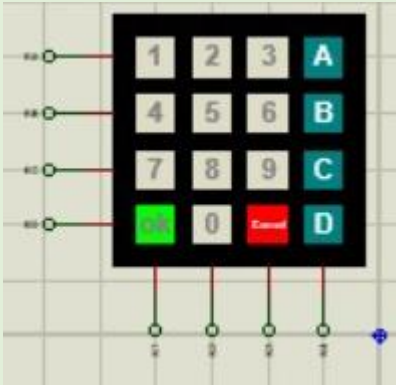
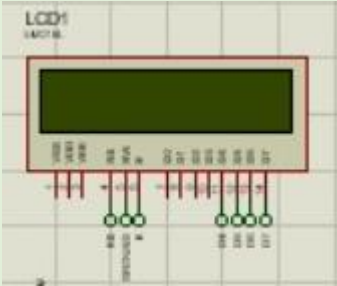
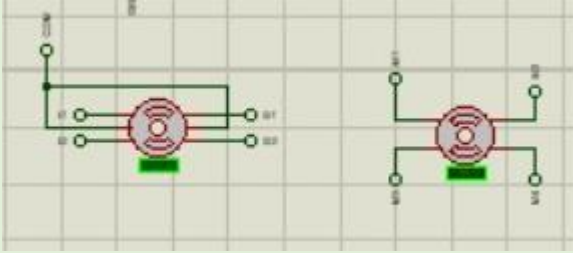

<p>Teclado alfanumérico matricial de 4x4</p>	
<p>Pantalla LCD 16x2</p>	
<p>Motores</p>	
<p>Fotorresistencias o LDR y Buzzer</p>	



Figura 3 – Pantalla de la Aplicación Android

Explicación de Código Arduino

```
Proyecto
#include <LedControl.h>
#include <EEPROM.h>
#include <LiquidCrystal.h>
#include <Keypad.h>
LiquidCrystal lcd(7, 6, 5, 4, 3, 2);
LedControl mc = LedControl(51, 53, 52, 1);

float LDR, LDR1, LDR2, LDR3, LDR4, LDR5, LDR6, LDR7, LDR8, LDR9, LDR10, LDR11, LDR12, LDR13, LDR14, LDR15;

byte candado[] = {
  B00100,
  B01010,
  B10001,
  B10001,
  B11111,
  B11011,
  B11111,
  B11111
};

int tenSeconds;
int buz = 9;
int led = 10;
char entrada;

/********* teclado *********/
const byte FILAS = 4;    // Filas
const byte COLUMNAS = 4; // Columnas
int estadoIngre = 0;
```

Figura 4 – Porción de código Arduino

```
Proyecto
void setup() {
  // put your setup code here, to run once:
  lcd.begin(16, 2); //16 Columnas 2 filas
  lcd.createChar(0, candado);
  lcd.clear();
  lcd.home();
  lcd.setCursor(2, 0);
  lcd.write(byte(0));
  lcd.setCursor(3, 0);
  lcd.print("Bienvenido");
  lcd.setCursor(13, 0);
  lcd.write(byte(0));
  lcd.setCursor(1, 1);
  lcd.print("Grupo5-SeccionA");
  Serial.begin(9600);
  Serial1.begin(9600);
  tenSeconds = -1;
  tokenConexion = "";
  tokenIngresado = "";
  randomSeed(analogRead(A0));
  intentosToken = 0;
  pinMode(buz, OUTPUT);
  pinMode(led, OUTPUT);
  parqueado = false;
  estado = 1;

  Serial.begin(9600);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
}
```

Figura 5 – Porción de código Arduino (setup)

```
void loop() {  
  if (tenSeconds == 11) { ///// Esperando Conexion  
    lcd.setCursor(0, 0);  
    lcd.print("Esperando");  
    lcd.setCursor(0, 1);  
    lcd.print("Conexion");  
    if (Serial.available() > 0) {  
      entrada = Serial.read();  
      if (entrada == 'C') {  
        lcd.clear();  
        lcd.setCursor(0, 0);  
        lcd.print("Conectado");  
        tenSeconds++;  
      }  
    }  
  }  
  else if (tenSeconds == 10) { /// solo me limpia la pantalla una vez  
    delay(1000);  
    lcd.clear();  
    tenSeconds++;  
  }  
  else if (tenSeconds == 12) { /// va esperar el login o registrar  
    if (Serial.available() > 0) {  
      entrada = Serial.read();  
      Serial.println(entrada);  
      if (entrada == 'l') { // login  
        lcd.clear();  
        delay(10);  
        login();  
      }  
      else if (entrada == 'r') { // registro
```

Figura 6 – Porción de código Arduino (loop)

CONCLUSIONES

- Es importante que al diseñar verifiquemos que los pines de entrada en el Arduino se utilicen una sola vez, ya que de lo contrario el circuito no funcionara como esperamos.
- Con los motores hay que tener especial cuidado para hacer los giros o para mantener el sentido de estos.
- Al momento de trabajar con el módulo bluetooth se debe validar siempre los dispositivos a los que se puede conectar,