UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
ARQUITECTURA DE COMPUTADORES Y ENSAMBLADORES 1
SECCIÓN B

Proyecto 1: Empresa Inteligente

Grupo #2

Diego René Molina Roldán 201314852
Yimmi Daniel Ruano Pernllo 201503470
Kevin Leonel Santizo Sosa 201504425

Guatemala, 14 de septiembre de 2020

Introducción

Este documento proporciona la descripción y base teórica sobre la cual se realizó la práctica, la manera en la que funcionan los motores del tipo stepper, tanto bipolares como unipolares, servomotores, modulo bluetooth y el uso de sensores, como el sensor de temperatura.

OBJETIVOS

Objetivo general

- Que el estudiante adquiera, aplique e interactúe con el microcontrolador Arduino.

Objetivos específicos

• Comprender el funcionamiento de las entradas y salidas, tanto digitales como análogas del microcontrolador Arduino.
• Comprender el funcionamiento de los dispositivos electromecánicos (motores DC, stepper, servomotores, etc.).
• Aplicar el uso de sensores de temperatura.
• Comprender la configuración de las pantallas lcd, los protocolos de comunicación en serio del arduino, I2C y UART
• Aplicar el lenguaje C para las estructuras de control en microcontroladores.

Marco Teórico

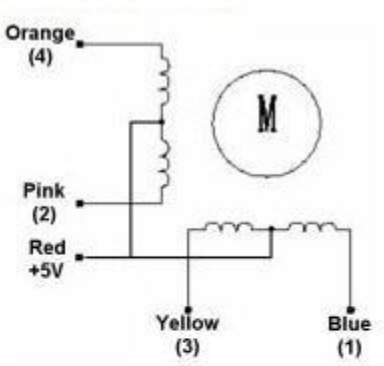
- Motor Stepper o Paso a Paso

Un motor paso a paso es un dispositivo electromecánico que convierte pulsos eléctricos en movimientos mecánicos discretos. El eje de un motor paso a paso gira en incrementos discretos cuando impulsos de mando eléctrico se aplican a él en la secuencia correcta.

La secuencia de los pulsos aplicados se relaciona directamente con la dirección de rotación de ejes motor. La velocidad de la rotación de los ejes motor está directamente relacionada con la frecuencia de los pulsos de entrada y la duración de la rotación está directamente relacionada con el número de pulsos de entrada aplicada.

Una de las ventajas más importantes de un motor paso a paso es su capacidad para ser controlado con precisión en un sistema de lazo abierto. Control de lazo abierto significa que ninguna información de retroalimentación de posición es necesario. Este tipo de control elimina la necesidad de costosos dispositivos de detección y regeneración como codificadores ópticos.

Uno de los motores stepper más usados en Arduino es el 28BYJ-48, el cual tiene un paso de 5.625 grados (64 pasos por vuelta usando half-step). El reductor interno tiene una relación de 1/64. Combinados, la precisión total es de 4096 pasos por vuelta, equivalente a un paso de 0.088º, que es una precisión muy elevada.



Half-Step Switching Sequence

Lead Wire Color	---> CW Direction (1-2 Phase)							
	1	2	3	4	5	6	7	8
4 Orange	-	-						-
3 Yellow		-	-	-				
2 Pink				-	-	-		
1 Blue						-	-	-

Imagen 1: Esquema de pasos del motor stepper

- Servomotor

Un servomotor (o servo) es un tipo especial de motor con características especiales de control de posición. Al hablar de un servomotor se hace referencia a un sistema compuesto por componentes electromecánicos y electrónicos.

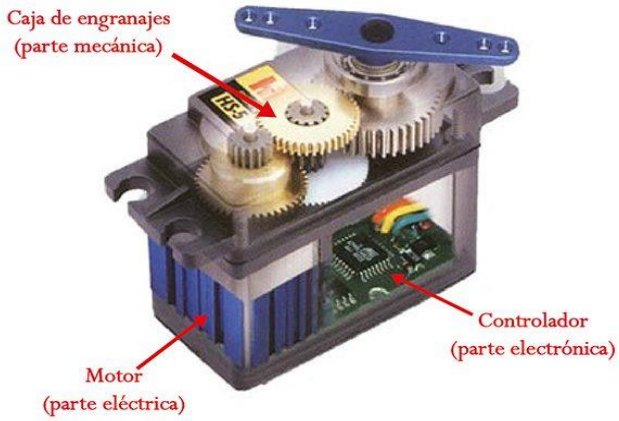


Imagen 2: Servomotor y separación de sus piezas

El motor en el interior de un servomotor es un motor DC común y corriente. El eje del motor se acopla a una caja de engranajes similar a una transmisión. Esto se hace para potenciar el torque del motor y permitir mantener una posición fija cuando se requiera. De forma similar a un automóvil, a menor mayor velocidad, menor torque. El circuito electrónico es el encargado de manejar el movimiento y la posición del motor.
La presencia del sistema de engranajes como el que se muestra en la figura hace que cuando movemos el eje motor se sienta una inercia muy superior a la de un motor común y corriente. Observando las imágenes que hemos presentado nos podemos dar cuenta que un servo no es un motor como tal, sino un conjunto de partes (incluyendo un motor) que forman un sistema.

Tipos de servomotores: Los servomotores de modelismo operan a voltajes bajos en corriente directa, típicamente entre 4 y 6 voltios. Los servomotores industriales operan tanto en DC como en AC (monofásico o trifásico). Para mí es un poco difícil escribir sobre este tipo de temas debido a que nunca he tenido la oportunidad de trabajar con un servo industrial. Lo que sé sobre ellos es por lo que he leído o lo habré visto en algún video o seminario. Los servos de modelismo, en cambio, se pueden adquirir a muy bajo costo en Internet y son populares entre los usuarios de Arduino. Se debe resaltar que, dentro de los diferentes tipos de servomotores, éstos se pueden clasificar según sus características de rotación.

Servomotores de rango de giro limitado: son el tipo más común de servomotor. Permiten una rotación de 180 grados, por lo cual son incapaces de completar una vuelta completa.

Servomotores de rotación continua: se caracterizan por ser capaces de girar 360 grados, es decir, una rotación completa. Su funcionamiento es similar al de un motor convencional, pero con las características propias de un servo. Esto quiere decir que podemos controlar su posición y velocidad de giro en un momento dado.
Los servomotores de rango de giro limitado se pueden adecuar para que funcionen como servomotores de rotación continua. Sin embargo, si requerimos un servo de 360 grados es mejor comprar uno que haya sido diseñado para este tipo de uso.

- Comunicación en serie

Los puertos serie son la forma principal de comunicar una placa Arduino con un ordenador. Gracias al puerto serie podemos, por ejemplo, mover el ratón o simular la escritura de un usuario en el teclado, enviar correos con alertas, controlar un robot realizando los cálculos en el ordenador, encender o apagar un dispositivo desde una página Web a través de Internet, o desde una aplicación móvil a través de Bluetooth.

Existen un sin fin de posibilidades en las que se requiere el empleo del puerto serie. Por tanto el puerto serie es un componente fundamental de una gran cantidad de proyectos de Arduino, y es uno de los elementos básicos que debemos aprender para poder sacar todo el potencial de Arduino.

En esta entrada aprenderemos el funcionamiento básico de los puertos serie en Arduino. Al final de la entrada se adjuntan varios códigos de ejemplo, pero antes conviene explicar brevemente algo de teoría sobre qué es un puerto serie, y algunos términos que necesitaremos para entender correctamente el funcionamiento del puerto serie.

Puedes consultar el resto de tutoriales sobre el puerto de serie en Arduino, (recibir números, textos, arrays separados por coma, bytes, y muchos más) en la categoría puerto serie Arduino

¿QUÉ ES EL PUERTO SERIE?
Un puerto es el nombre genérico con que denominamos a los interfaces, físicos o virtuales, que permiten la comunicación entre dos ordenadores o dispositivos.

Un puerto serie envía la información mediante una secuencia de bits. Para ello se necesitan al menos dos conectores para realizar la comunicación de datos, RX (recepción) y TX (transmisión). No obstante, pueden existir otros conductores para referencia de tensión, sincronismo de reloj, etc.

Por el contrario, un puerto paralelo envía la información mediante múltiples canales de forma simultánea. Para ello necesita un número superior de conductores de comunicación, que varían en función del tipo de puerto. Igualmente existe la posibilidad de conductores adicionales además de los de comunicación.

- Puerto serie

Históricamente ambos tipos de puertos han convivido en los ordenadores, empleándose los puertos paralelos en aquellas aplicaciones que requerían la transmisión de mayores volúmenes de datos. Sin embargo, a medida que los procesadores se hicieron más rápidos los puertos de serie fueron desplazando progresivamente a los puertos paralelos en la mayoría de aplicaciones.

Un ordenador convencional dispone de varios puertos de serie. Los más conocidos son el popular USB (universal serial port) y el ya casi olvidado RS-232 (el de los antiguos ratones). Sin embargo, dentro del ámbito de la informática y automatización existen una gran cantidad adicional de tipos de puertos serie, como por ejemplo el RS-485, I2C, SPI, Serial Ata, Pcie Express, Ethernet o FireWire, entre otros.

En ocasiones veréis referirse a los puertos de serie como UART. La UART (universally asynchronous receiver/transmitter) es una unidad que incorporan ciertos procesadores, encargada de realiza la conversión de los datos a una secuencia de bits y transmitirlos o recibirlos a una velocidad determinada.

Por otro lado, también podéis oír el término TTL (transistor-transistor logic). Esto significa que la comunicación se realiza mediante variaciones en la señal entre 0V y Vcc (donde Vcc suele ser 3.3V o 5V). Por el contrario, otros sistemas de transmisión emplean variaciones de voltaje de -Vcc a +Vcc (por ejemplo, los puertos RS-232 típicamente varían entre -13V a 13V).

Antes de conectar dos sistemas debemos comprobar que los voltajes empleados son compatibles. En caso de no serlo, necesitaremos un subsistema que adapte los niveles de la señal, o podemos dañar alguno de los dispositivos.

ARDUINO Y EL PUERTO SERIE

Prácticamente todas las placas Arduino disponen al menos de una unidad UART. Las placas Arduino UNO y Mini Pro disponen de una unidad UART que operan a nivel TTL 0V / 5V, por lo que son directamente compatibles con la conexión USB. Por su parte, Arduino Mega y Arduino Due disponen de 4 unidades UART TTL 0V / 5V.

Los puertos serie están físicamente unidos a distintos pines de la placa Arduino. Lógicamente, mientras usamos los puertos de serie no podemos usar como entradas o salidas digitales los pines asociados con el puerto serie en uso.

En Arduino UNO y Mini Pro los pines empleados son 0 (RX) y 1 (TX). En el caso de Arduino Mega y Arduino Due, que tienen cuatro puertos de serie, el puerto serie 0 está conectado a los pines 0 (RX) y 1 (TX), el puerto serie 1 a los pines 19 (RX) y 18 (TX) el puerto serie 2 a los pines 17 (RX) y 16 (TX), y el puerto serie 3 a los pines 15 (RX) y 14 (TX).

Muchos modelos de placas Arduino disponen de un conector USB o Micro USB conectado a uno de los puertos de serie, lo que simplifica el proceso de conexión con un ordenador. Sin embargo algunas placas, como por ejemplo la Mini Pro, prescinden de este conector por lo que la única forma de conectarse a las mismas es directamente a través de los pines correspondientes.

- Sensor LM35

El LM35 es un sensor de temperatura digital. A diferencia de otros dispositivos como los termistores en los que la medición de temperatura se obtiene de la medición de su resistencia eléctrica, el LM35 es un integrado con su propio circuito de control, que proporciona una salida de voltaje proporcional a la temperatura.
La salida del LM35 es lineal con la temperatura, incrementando el valor a razón de 10mV por cada grado centígrado. El rango de medición es de -55ºC (-550mV) a 150ºC (1500 mV). Su precisión a temperatura ambiente es de 0,5ºC.
Los sensores LM35 son relativamente habituales en el mundo de los aficionados a la electrónica por su bajo precio, y su sencillez de uso.

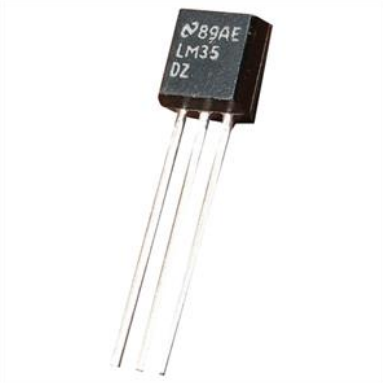


Imagen 3: Sensor LM35

ESQUEMA ELÉCTRICO

El patillaje del LM35 se muestra en la siguiente imagen. Los pines extremos son para alimentación, mientras que el pin central proporciona la medición en una referencia de tensión, a razón de 10mV/ºC.

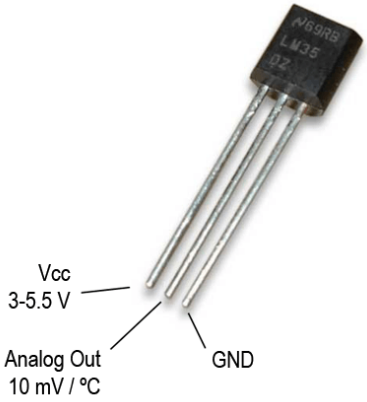


Imagen 4: Esquema eléctrico del sensor LM35

- LCD I2C

El controlador de LCD I2C es un dispositivo que nos permite controlar una pantalla a través del bus I2C, usando únicamente dos cables.
En esta entrada aprendimos a manejar un display LCD Hitachi con controlador HD44780, una familia de pantallas barata y sencillas de emplear.
Pero usar esta pantalla directamente desde Arduino requería emplear una gran cantidad de pines de Arduino, lo que supone un enorme desperdicio de recursos, que deberían estar ocupados en cosas mucho más importantes que encender un simple display.
Una alternativa recomendable es usar un controlador que permita acceder al LCD a través del bus I2C. Este controlador LCD I2C puede conectarse a cualquier LCD Hitachi HD44780 y reduce la cantidad de cables necesarios a dos.
Internamente el controlador LCD I2C es una variación del extensor de entradas y salidas digitales PCF8574, especialmente adaptado para pantallas LCD Hitachi HD44780. Incluso incorporan un potenciómetro para regular el backlight del LCD.
El controlador LCD I2C normalmente se entrega por separado, en cuyo caso tendremos que soldarlo al display LCD.

Imagen 5: Controlador LCD I2C y pantalla LCD

Esquema

La conexión es sencilla, simplemente alimentamos el módulo desde Arduino mediante GND y 5V y conectamos el pin SDA y SCL de Arduino con los pines correspondientes del controlador LCD I2C.

Esquema de montaje

Para el diseño del circuito se hace uso de los siguientes elementos electrónicos:

Elemento	Cantidad
Placa Arduino Mega	1
Motores DC	2
CI L297	1
Leds	8
Motores Stepper	2
Sensores Temperatura	1
LCD LM016L	1
Speaker	1
Modulo bluetooth HC-06	1
CI PCF8574	1
Keypad customizado	1

Tabla 1: Cuantificación de elementos

Descripción del problema

Descripción:

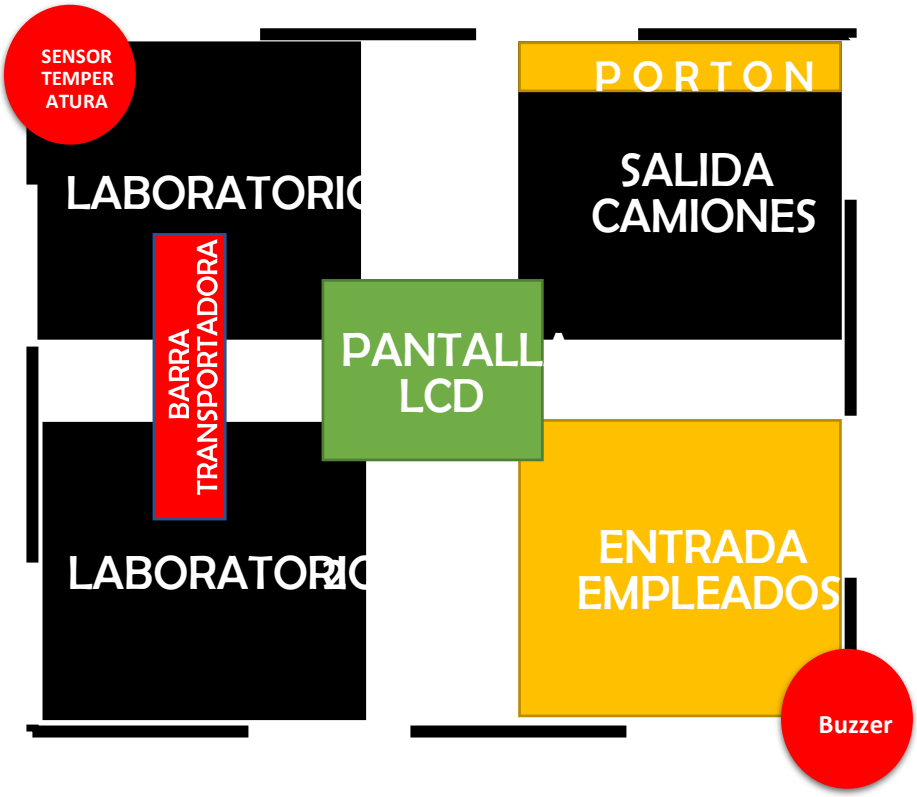
Una empresa de paquetería desea automatizar sus procesos, desde el control de entrada de sus empleados, control de sus procesos y hasta el control de sus luces y servicios, y les pide a ustedes como empresa independiente de IOT el modelo de un sistema para su empresa, donde todo podrá ser controlado mediante una aplicación móvil conectada a todo lo que sea posible de esta por medio de bluetooth y el microcontrolador arduino.

La empresa desea dicho modelo en la plataforma "Proteus" para así poder ver el resultado final mediante una simulación en esta aplicación y decidir si ustedes son el equipo correcto para llevar a cabo lo que desean.

Los procesos que la empresa desea poder controlar son:

Procesos a controlar:

- Entrada de los empleados a la Empresa por medio de una clave.
- Control de sus luces en distintos escenarios.
- Control de barra transportadora.
- Control de Porton.
- Control Temperatura de toda la empresa.
- Control de su Alarma General



1. Entrada de los empleados a la Empresa por medio de una clave:

Al iniciar la simulación se motrará un mensaje en la pantalla lcd de bienvenida, por 5 segundos, el mensaje a mostrar es:

A.

Pasados los 5 segundos, se le pedira al usuario el ingreso de su número de identificación y su clave, y se tendrá la secuencia de la imagen 1

El primer paso para realizar como usuario será el de ingresar a la empresa por medio de una clave y numero de identificación, la cual será ingresada por medio de un *Keypad.*

Además, si el usuario no posee una clave y número de identificación deberá escribir con el teclado 0000 para entrar al modo de registrar usuario.

Al entrar a este modo deberá ingresar su clave que no exceda de los 8 dígitos.

- Si la clave excede de los 8 dígitos deberá de mostrarse un error.

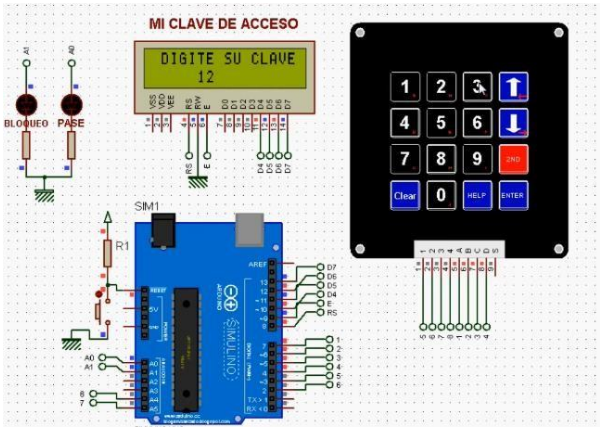
Luego de ingresar su clave se le pedirá que la ingrese nuevamente. ○ Si las claves ingresadas coinciden deberá de mostrar un error.

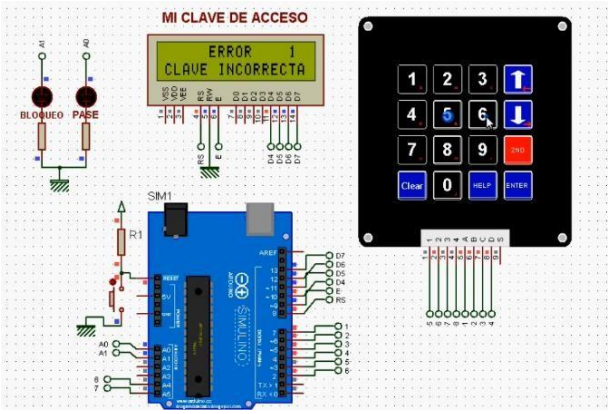
Posteriormente le pedirá que el gerente ingrese su clave para validar la operación.

Por último, si todos los pasos son correctos se le proporcionara en la pantalla LCD el número que lo identifica.

NOTA: Tanto el identificador como la contraseña deben de ser almacenados en la memoria EEPROM.

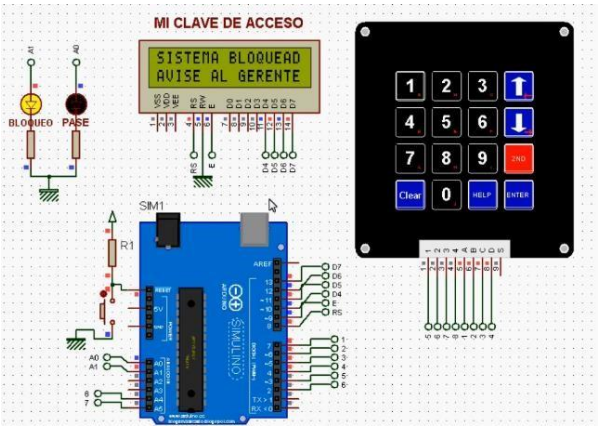
Luego de que el usuario posea un número de identificación y contraseña podrá ingresar al sistema. Estos podrían ser ingresados erroneamente una cantidad de 3 veces, si se ingresa una 4ta vez de manera errónea, la empresa entrara en un estado de desbloqueo y se deberá notificar al gerente de la empresa para el desbloqueo de esta, el cual será por medio de una clave de 4 dígitos: **Clave Gerente:** *0106*





Si la clave es ingresada erroneamente 4 veces

Si la clave es ingresada correctamente



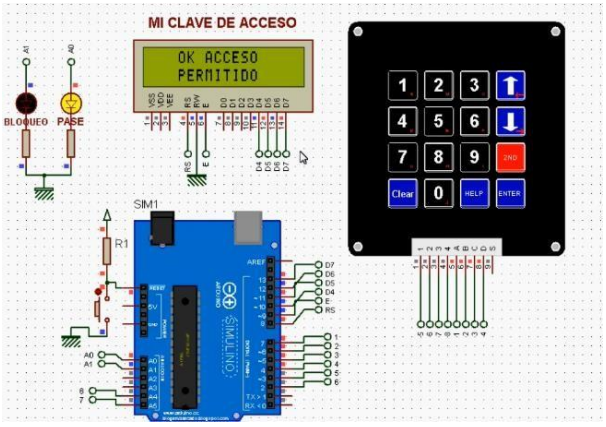


Imagen 1

Credenciales de usuario ingresados correctamente:
Al ser ingresada correctamente las credenciales del empleado, en el rango de las 4 oportunidades:

- se mostrara un mensaje en la pantalla lcd de acceso permitido.
- se generara un tono de 2 segundos con el buzzer.
- Se encendera un led permitiendo el paso del empleado.
- Todas las luces de la empresa se encenderan.
- Se permitira la conexión de la aplicación móvil con el Arduino.
  Credenciales de usuario ingresada erróneamente 4 veces:

Al ser ingresada erróneamente las credenciales del empleado una cantidad de 4 veces:

- se mostrará un en la pantalla lcd mensaje de “sistema bloqueado, avise al gerente”.
- se generara un tono de 5 segundos con el buzzer.
- se pedira el ingreso de la clave del gerente.
- se pedira nuevamente la clave del empleado.

Nota: **No se debe permitir la conexión al arduino por medio del bluetooth hasta que las credenciales del empleado se haya ingresado correctamente.**

Led **Acceso Permitido : Color Amarillo**

Led Sistema Bloqueado : Color Rojo

2. Control de barra transportadora.

La barra transportadora, es aquella que comunica al laboratorio 1 con el laboratorio 2, y transporta paquetes en ambas direcciones de estos laboratorios, es decir un paquete podrá ir desde el labotarorio 1 hacia el laboratorio 2 y viceversa.

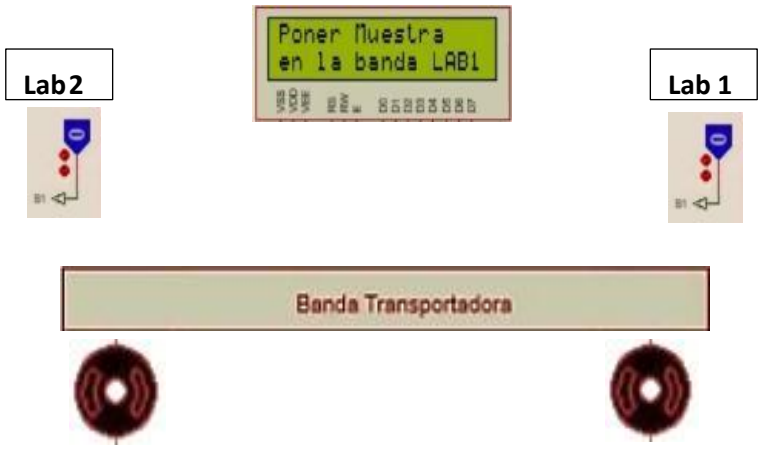
La banda estará conformada por dos motores stepper, el cúal uno podra ser controlado por medio de un modulo/driver extra, a elección de los estudiantes, y otro no, los paquetes seran simulados por status label, uno en cada laboratorio:

Status Label = 0 , no hay paquete en laboratorio, Status Label = 1 hay paquete en ese laboratorio.

La banda será iniciada por medio de la app móvil, la cual en la seccion de banda contara con 2 botones, uno por laboratorio, al presionar el boton de X laboratorio, la banda tendrá que iniciar con dirección hacia el otro laboratorio, **pero**, debe tomar en cuenta que si el status label indica que en el laboratorio X no hay un paquete, está no iniciará y mostrara un msj en la pantalla lcd indicando que se cargue un paquete en el laboratorio X, al ser cargado dicho paquete, esta iniciara, y al indicar el status label X que ya no hay paquete (estado =0) y que en el label del otro laboratorio ya se encuentra el paquete (estado = 1) la banda deberá parar.

Todo este proceso debe ser validado para ambos laboratorios.

El boton de laboratorio 1 es encendido, pero ningun paquete a sido cargado del lado de este laboratorio.



El paquete es colocado en el lab 1 y el boton de este es vuelto a presionar, por lo que la banda inicia, yendo hacia el lab 2 (motores giran a la izquierda).

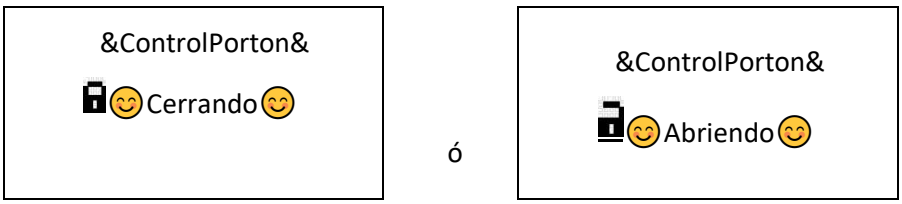
El paquete ha llegado al laboratorio 2, por lo que se muestra un mensaje en la pantalla lcd, y la banda transportadora

Al momento de encender la banda transportadora, el buzzer debe generar un tono por 3 seg. Y cada que un paquete llegue a un laboratorio se debe generar tambien un tono (distinto por laboratorio) de 1 seg.

3. Control de Portón

La aplicación móvil contara con la sección de control de portón, donde simplemente podrá abrir o cerrar el portón por donde salen los camiones de la empresa, este será simulado por medio de un servomotor, el cual para abrir girara 2 vueltas hacia la derecha, y al terminar dicho movimiento se encenderá un led de color rojo, el portón permanece de esa manera durante 6 segundos, pasado el tiempo se cerrara automáticamente, si es que el botón de cerrar no es presionado antes, para cerrar el motor girara de manera contraria otras 2 vueltas, indicando el final del cierre con un led amarillo y un tono producido por el buscar de 3 seg.

Durante el control del portón se debe indicar un mensaje en la pantalla cd de la siguiente manera:



4. Control de sus luces en distintos escenarios.

Cada sección de la empresa contará con un led (simulando las luces de esta) de color azul

Secciones:

- Laboratorio 1
- Laboratorio 2
- Entrada Empleados
- Salida Camiones

Y estas serán controladas de la siguiente manera:

- Deben encenderse todas al accesar a la empresa
- Podrán ser apagadas o encendidas desde la aplicación móvil, ya sea una por una ó todas al mismo tiempo.

5. Control Temperatura de toda la empresa.

La aplicación móvil contara con la sección de “Lectura de Temperatura”, la cual al ingresar a ella, se activara un sensor de temperatura, el cual hará la lectura correspondiente de está y le enviara al usuario por medio del bluetooth un mensaje indicando la temperatura actual de la empresa, está será mostrada únicamente en la aplicación móvil.

6. Control de su Alarma General

La alarma general de la empresa son los tonos producidos por el buzzer en las distintas situaciones indicadas, cada tono debe ser distinto de los demás y ser producido por el tiempo indicado.

7. Flujo aplicación Móvil

Se Iniciará la aplicación móvil, pidiendo al usuario que se conecte al dispositivo.
Se mostrará un menú general al usuario con las siguientes secciones:

- **Control Banda:** Contará con dos botones, uno por laboratorio, los cuales iniciaran la banda, como se indicó anteriormente.
- **Control Portón** Contará con dos botones: abrir portón, cerrar portón.
- **Control Luces** Contará con dos botones por sección para apagar o encender individualmente cada luz, y con dos botones extras para el apagado y encendido general de la luces.
- **Control Temperatura** Se mostrará la lectura del sensor de temperatura.

Diagrama del circuito

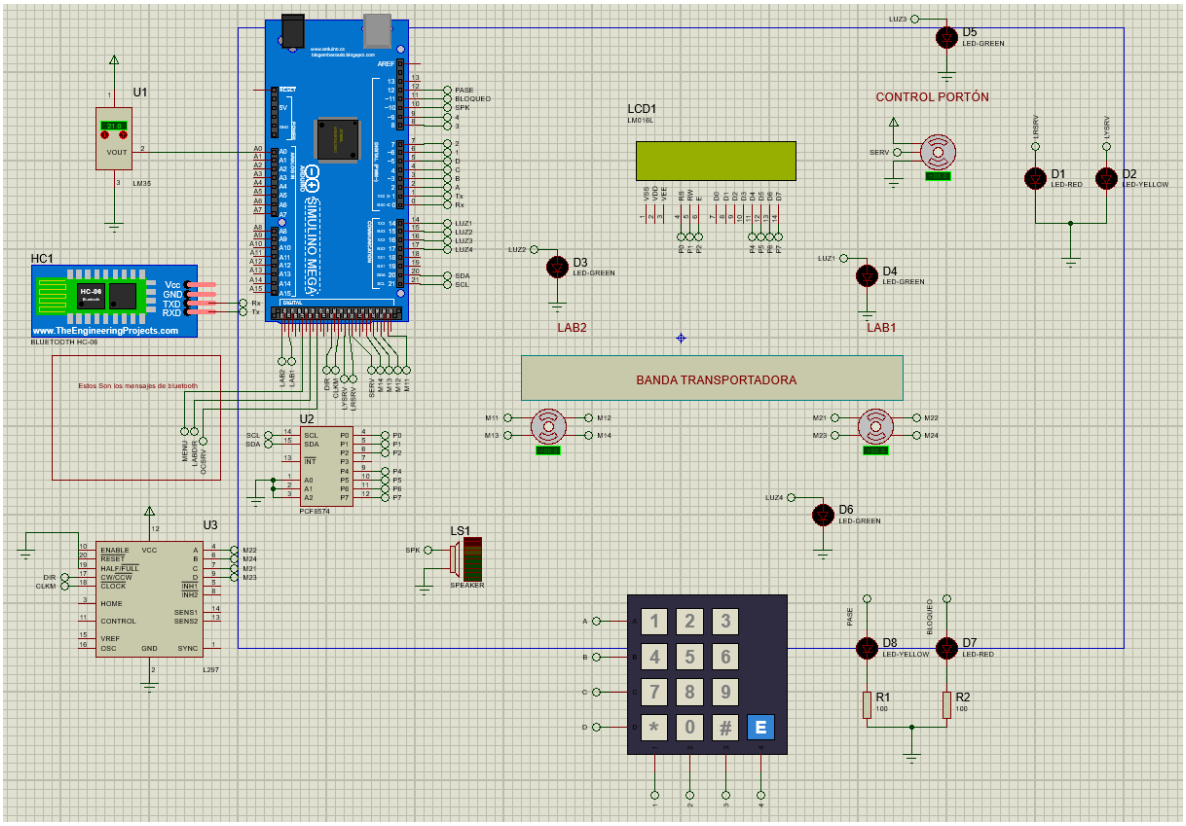


Imagen 4: Diagrama general del circuito

Código realizado para el microcontrolador

```
#include <Wire.h>
#include "LiquidCrystal_I2C.h"
//INCLUIDO MOTORES ------------------------------------------------
#include <Servo.h>
#include <Stepper.h>

#include <Keypad.h>
#include <EEPROM.h>

struct EMPLEADO {
  char ID[10];
  char PSW[10];
};

//Entradas
#define INLAB1 51
```

```
#define INLAB2 53

LiquidCrystal_I2C lcd (0x20, 16, 2);

//VARIABLES MOTORES-------------------------------------------------
Servo srv;

const int MST1[4] = {23, 25, 27, 29};

const int patron[4][4] = {
  {1, 1, 0, 0},
  {0, 1, 1, 0},
  {0, 0, 1, 1},
  {1, 0, 0, 1},
};

//Mis variables:
byte lock[8]  = {0xe, 0x11, 0x11, 0x1f, 0x1b, 0x1b, 0x1f};
byte unlock[8]  = {0xe, 0x1, 0x1, 0x1f, 0x1b, 0x1b, 0x1f};
byte smyle[8]  = {0x0, 0xa, 0xa, 0x0, 0x11, 0x0e};
byte checked[8] = {0x1, 0x2, 0x16, 0x1c, 0x8};

int slab1 = 0;
int slab2 = 0;
int pospa = 0;
int clkst2 = 0;
int cerrarPuerta = 0;
unsigned long t = 0;
unsigned long t2 = 0;
unsigned long t3 = 0;
unsigned long t8 = 0;
boolean acces = false;
boolean cerrado = true;

//EL SIGUIENTE SEGMENTO SE PUEDE ELIMINAR:--------------------------&&&&
#define INMENU 47
#define INLABDIR 45
#define OPCLSRV 43
int menu = 0;
int slabdir = 0;
//-----------------------------------------------------------------&&&&


char state;   // Variable lectrura dato serial
```

```
#define pinTemp A0
int temperatura = 0;
float temp;

int ledPASE = 12;
int ledBLOQUEO = 11;
const byte ROWS = 4;
const byte COLS = 4;
int buzzer = 10;
//CCONTADOR DE INTENTOS DE ACCESO
int contadorIntento = 0;
char keys[ROWS][COLS] = {
  {'1', '2', '3', '-'},
  {'4', '5', '6', '-'},
  {'7', '8', '9', '-'},
  {'*', '0', '#', 'E'}
};
byte rowPins[ROWS] = {2, 3, 4, 5};
byte colPins[COLS] = {6, 7, 8, 9};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
byte candado [8] = { B00000, B01110, B01010, B11111, B10001, B10001, B11111,
 B00000};
int cantempleados;

//ESTADOS DE ACCESO AL SISTEMA
boolean estadoMsg = true;
boolean estadoID = false;
boolean estadoContrasena = false;
boolean estadoFallido = false;
boolean estadoBloqueado = false;
boolean estadoCorrecto = false;
boolean estadoRegistro = false;
boolean estadoDesbloquear = false;

//VARIABLES TEMPORALES DE ID Y PSW INGRESADOS
String ID = "";
String PSW = "";
String EID = "";
String EPSW = "";
String EGRT = "";


void setup() {
  // put your setup code here, to run once:
  //MOTOR 1 STEPER
```

```
  pinMode(23, OUTPUT);
  pinMode(25, OUTPUT);
  pinMode(27, OUTPUT);
  pinMode(29, OUTPUT);
  //MOTOR 2 STEPPER
  //RELOJ 2 STEP
  pinMode(37, OUTPUT);
  //DIRECCION 2 STEP
  pinMode(39, OUTPUT);
  //LEDS
  pinMode(34, OUTPUT);
  pinMode(35, OUTPUT);
  //LAB1
  pinMode(INLAB1, INPUT);
  //LAB2
  pinMode(INLAB2, INPUT);
  //LUCES
  pinMode(14, OUTPUT);
  pinMode(15, OUTPUT);
  pinMode(16, OUTPUT);
  pinMode(17, OUTPUT);

  //SENSOR TEMPERATURA
  pinMode(A0, INPUT);



  //EL SIGUIENTE SEGMENTO SE PUEDE ELIMINAR, CUANDO ESTE LO DE BLUETOOTH:---
-----------------------&&&&
  //LABDIR
  pinMode(INLABDIR, INPUT);
  //MENU
  pinMode(INMENU, INPUT);
  //CerrarSRV
  pinMode(OPCLSRV, INPUT);
  //-------------------------------------------------------------------&&&&


  lcd.init();
  lcd.backlight();


  lcd.createChar(2, lock);
  lcd.createChar(3, unlock);
  lcd.createChar(4, smyle);
  lcd.createChar(5, checked);
```

```
  srv.attach(33);
  srv.write(0);
  t = millis();
  t2 = millis();
  t8 = millis();

  cerrado = true;
  Serial.begin(9600);

  pinMode(ledPASE, OUTPUT);
  pinMode(ledBLOQUEO, OUTPUT);
  lcd.init();
  lcd.backlight();
  lcd.createChar(1, candado);


  //LIMPIAR PRIMEROS 200 BYTES
  /*for (int i = 0 ; i < 200 ; i++) {
    EEPROM.write(i, 0);
    Serial.println(i);
    }*/

  //Borrando los empleados
  //cantempleados = 0;
  //EEPROM.put(0, cantempleados);
  //EEPROM.get(0, cant);

  int seed;
  EEPROM.get(0, seed);
  randomSeed(seed + 32);

}


boolean login_state;

int period = 1000;
unsigned long time_now = 0;

void loop() {

     if (cerrado == false && millis() - t3 >= 6000) {
            cerrarPorton();
            cerrado = true;
     }
```

```
login_state = LOGIN();
if (!login_state) {
  //Y se envia cuando no hay acceso a la app

  if (millis() > time_now + period) {
    time_now = millis();
    Serial.print("Y");
  }
  return;
} else {
  Serial.print("");
  if (Serial.available() > 0) {
    state = Serial.read();
    switch (state) {
      case 'A':
        digitalWrite(14, HIGH);
        state = 0;
        break;
      case 'B':
        digitalWrite(14, LOW);
        state = 0;
        break;
      case 'C':
        digitalWrite(15, HIGH);
        state = 0;
        break;
      case 'D':
        digitalWrite(15, LOW);
        state = 0;
        break;
      case 'E':
        digitalWrite(16, HIGH);
        state = 0;
        break;
      case 'F':
        digitalWrite(16, LOW);
        state = 0;
        break;
      case 'G':
        digitalWrite(17, HIGH);
        state = 0;
        break;
      case 'H':
        digitalWrite(17, LOW);
```

```
        state = 0;
        break;
      case 'I':
        digitalWrite(14, HIGH);
        digitalWrite(15, HIGH);
        digitalWrite(16, HIGH);
        digitalWrite(17, HIGH);
        state = 0;
        break;
      case 'J':
        digitalWrite(14, LOW);
        digitalWrite(15, LOW);
        digitalWrite(16, LOW);
        digitalWrite(17, LOW);
        state = 0;
        break;
      case 'K':
        slabdir = 0;
        fun_barra_trans();
        state = 0;
        break;
      case 'L':
        slabdir = 1;
        fun_barra_trans();
        state = 0;
        break;
      case 'M':
        cerrarPuerta = 0;
        abrirPorton();
        t3 = millis();
        cerrado = false;
        state = 0;
        break;
      case 'N':
        cerrarPuerta = 1;
        cerrarPorton();
        cerrado = true;
        state = 0;
        break;
      case 'T':
        temperatura = analogRead(A0);
        temp = temperatura / 2;
        Serial.print(temp);
        state = 0;
    }
```

```
    }
    //Eliminar el digitalRead, a que se lea el Bluetooth. el switch puede se
guir, el tipo de dato menu es int, cambiar a char.
  }
}

void fun_barra_trans() {

  //VEO HACIA QUE DIRECCION TIENE QUE IR LA CARGA, ESTE IF, PUEDE IR EN LA S
ECCION DEL SWITCH DEL LOOP,
  //PARA DESAPARECER ESTE METODO Y QUE SOLO SE DECIDA SI HAY QUE IR A LOS ME
TODOS Lab1_to_Lab2(); o al otro.


  slab1 = digitalRead(INLAB1);
  slab2 = digitalRead(INLAB2);
  if (slabdir == 0) {
    //VA DE LAB1 A LAB2
    Lab1_to_Lab2();
  } else {
    //VA DE LAB2 A LAB1
    Lab2_to_Lab1();
  }
}

void Lab1_to_Lab2() {
  if (slab1 == 1 && slab2 == 0) {
    bool salir = false;
    bool sono = false;
    while (!salir) {
      lcd.setCursor(0, 0);
      lcd.print("Corriendo hacia ");
      lcd.setCursor(0, 1);
      lcd.print("la izquierda    ");
      funStepIzq();
      slab1 = digitalRead(INLAB1);
      slab2 = digitalRead(INLAB2);
      if (!sono) {
        tone(10, 100, 3000);
        sono = true;
      }
      if (slab1 == 0 && slab2 == 1) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("La muestra llego");
```

```
        lcd.setCursor(0, 1);
        lcd.print("al LAB 2        ");
        tone(10, 100, 1000);
        delay(1500);
        salir = true;
      }
    }
  } else {
    lcd.setCursor(0, 0);
    lcd.print("Poner Muestra en");
    lcd.setCursor(0, 1);
    lcd.print("la banda LAB1   ");
  }
}

void Lab2_to_Lab1() {
  if (slab1 == 0 && slab2 == 1) {
    bool salir = false;
    bool sono = false;
    while (!salir) {
      lcd.setCursor(0, 0);
      lcd.print("Corriendo hacia ");
      lcd.setCursor(0, 1);
      lcd.print("la derecha     ");
      funStepDer();
      slab1 = digitalRead(INLAB1);
      slab2 = digitalRead(INLAB2);
      if (!sono) {
        tone(10, 800, 3000);
        sono = true;
      }
      if (slab1 == 1 && slab2 == 0) {
        lcd.clear();
        lcd.setCursor(0, 0);
        lcd.print("La muestra llego");
        lcd.setCursor(0, 1);
        lcd.print("al LAB 1        ");
        tone(10, 800, 1000);
        delay(1500);
        salir = true;
      }
    }
  } else {
    lcd.setCursor(0, 0);
    lcd.print("Poner Muestra en");
```

```
    lcd.setCursor(0, 1);
    lcd.print("la banda LAB2   ");
  }
}

void funStepIzq() {
  for (int j = 0; j < 4; j++) {
    if (patron[pospa][j] == 1) {
      digitalWrite(MST1[j], HIGH);
    } else {
      digitalWrite(MST1[j], LOW);
    }
  }
  digitalWrite(39, HIGH);
  if (millis() - t2 >= 65) {
    if (clkst2 == 0) {
      digitalWrite(37, HIGH);
      clkst2 = 1;
    } else {
      digitalWrite(37, LOW);
      clkst2 = 0;
    }
    t2 = millis();
  }
  if (millis() - t >= 130) {
    pospa--;
    if (pospa == -1) {
      pospa = 3;
    }
    t = millis();
  }
}

void funStepDer() {
  for (int j = 0; j < 4; j++) {
    if (patron[pospa][j] == 1) {
      digitalWrite(MST1[j], HIGH);
    } else {
      digitalWrite(MST1[j], LOW);
    }
  }
  digitalWrite(39, LOW);
  if (millis() - t2 >= 65) {
    if (clkst2 == 0) {
      digitalWrite(37, HIGH);
```

```
      clkst2 = 1;
    } else {
      digitalWrite(37, LOW);
      clkst2 = 0;
    }
    t2 = millis();
  }
  if (millis() - t >= 130) {
    pospa++;
    if (pospa == 4) {
      pospa = 0;
    }
    t = millis();
  }
}

//FUNCION DEL PORTON
void abrirPorton() {
  if(cerrado){
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.write(byte(5));
    lcd.setCursor(1, 0);
    lcd.print("&ControlPorton&");
    lcd.setCursor(2, 1);
    lcd.write(byte(3));
    lcd.setCursor(3, 1);
    lcd.write(byte(4));
    lcd.setCursor(4, 1);
    lcd.print("Abriendo");
    lcd.setCursor(12, 1);
    lcd.write(byte(4));
    for (int i = 0; i <= 180; i++) {
      srv.write(i);
      delay(15);
    }
    digitalWrite(34, HIGH);
  }
}

void cerrarPorton() {
  if(!cerrado) {
    digitalWrite(34, LOW);
    lcd.clear();
    lcd.setCursor(0, 0);
```

```
    lcd.write(byte(5));
    lcd.setCursor(1, 0);
    lcd.print("&ControlPorton&");
    lcd.setCursor(2, 1);
    lcd.write(byte(2));
    lcd.setCursor(3, 1);
    lcd.write(byte(4));
    lcd.setCursor(4, 1);
    lcd.print("Cerrando");
    lcd.setCursor(12, 1);
    lcd.write(byte(4));
    for (int i = 180; i >= 0; i--) {
      srv.write(i);
      delay(15);
    }
    digitalWrite(35, HIGH);
    tone(10, 550, 3000);
    delay(3000);
    digitalWrite(35, LOW);
  }
}



boolean LOGIN() {



  //ESTADO DE MENSAJE DE BIENVENIDA
  if (estadoMsg) {
    msg_bienvenida();
    estadoMsg = false;
    estadoID = true;
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("IDENTIFICADOR:");
    lcd.setCursor(6, 1);
  }

  //ESTADO DE INGRESO DE ID
  if (estadoID) {
    char key = keypad.getKey();
    if (key) {
      if (key == 'E') {
        if (ID == "0000") {
```

```
        ID = "";
        estadoID = false;
        estadoRegistro = true;
      } else {
        estadoID = false;
        lcd.clear();
        lcd.setCursor(2, 0);
        lcd.print("CONTRASENA:");
        lcd.setCursor(4, 1);
        estadoContrasena = true;
      }


    } else {
      lcd.print(key);
      ID = ID + key;
    }
  }
}

//ESTADO INGRESO DE CONTRASEÑA
if (estadoContrasena) {
  char key = keypad.getKey();
  if (key) {
    if (key == 'E') {
      estadoContrasena = false;
      if (ValidacionDatos(ID, PSW)) {
        estadoCorrecto = true;
      } else {
        estadoFallido = true;
      }
      ID = "";
      PSW = "";
      lcd.clear();
    } else {
      lcd.print(key);
      PSW  = PSW  + key;
    }
  }
}

//ESTADO ERROR (INTENTELO DE NUEVO)
if (estadoFallido) {
  contadorIntento++;
  lcd.setCursor(5, 0);
```

```
    lcd.print("ERROR");
    lcd.setCursor(15, 0);
    lcd.print(contadorIntento);
    lcd.setCursor(0, 1);
    lcd.print("ACCESO DENEGADO");
    delay(2000);
    if (contadorIntento == 4) {
      lcd.clear();
      estadoBloqueado = true;
    } else {
      lcd.clear();
      lcd.setCursor(1, 0);
      lcd.print("IDENTIFICADOR:");
      lcd.setCursor(6, 1);
      estadoID = true;
    }
    estadoFallido = false;
  }

  //ESTADO MODO BLOQUEO
  if (estadoBloqueado) {
    lcd.setCursor(1, 0);
    lcd.print("SIST BLOQUEADO");
    lcd.setCursor(0, 1);
    lcd.print("AVISE AL GERENTE");
    digitalWrite(ledBLOQUEO, HIGH);
    for (int i = 0; i <= 5; i++) {
      SPKBLOQUEO();
    }
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("CLAVE GERENTE:");
    lcd.setCursor(6, 1);
    estadoBloqueado = false;
    estadoDesbloquear = true;
  }

  //ESTADO DESBLOQUEAR
  if (estadoDesbloquear) {
    char key = keypad.getKey();
    if (key) {
      if (key == 'E') {
        if (EGRT == "0106") {
          lcd.clear();
          lcd.setCursor(1, 0);
```

```
          lcd.print("IDENTIFICADOR:");
          lcd.setCursor(6, 1);
          estadoID = true;
          contadorIntento = 0;
          estadoDesbloquear = false;
          digitalWrite(ledBLOQUEO, LOW);
        } else {
          estadoBloqueado = true;
          estadoDesbloquear = false;
          lcd.clear();
        }
        EGRT = "";
      } else {
        lcd.print(key);
        EGRT = EGRT  + key;
      }
    }
  }


  //ESTADO DE REGISTRO DE EMPLEADO
  if (estadoRegistro) {
    Registro();
  }
  //ESTADO CREDENCIALES CORRECTAS


  if (estadoCorrecto) {
    acces = true;
    lcd.setCursor(5, 0);
    lcd.print("ACCESO");
    lcd.setCursor(4, 1);
    lcd.print("PERMITIDO");
    digitalWrite(ledPASE, HIGH);
    estadoCorrecto = false;
    for (int i = 0; i <= 5; i++) {
      SPKBIENVENIDO();
    }

  }


  return acces;
}
```

```
int estadoregistro = 0;
String nuevoID = "";
String nuevaPsw = "";
String repeatPsw = "";
String adminPsw = "";


void Registro() {

  //MOSTRANDO MENSAJE NUEVA CONTRASEÑA
  if (estadoregistro == 0) {
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("NUEVA CONTRASENA");
    lcd.setCursor(4, 1);
    estadoregistro = 1;
  }

  //INTRODUCIENDO CLAVE NUEVA
  if (estadoregistro == 1) {

    char key = keypad.getKey();
    if (key) {
      if (key == 'E') {
        if (nuevaPsw.length() <= 8) {
          estadoregistro = 2;
        } else {
          estadoregistro = 7;
        }
        lcd.clear();
      } else {
        lcd.print(key);
        nuevaPsw = nuevaPsw  + key;
      }
    }
  }

  //MOSTRANDO MENSAJE DE CONFIRMACION DE CONTRASEÑA
  if (estadoregistro == 2) {

    lcd.clear();
    lcd.setCursor(2, 0);
    lcd.print("CONFIRMACION");
    lcd.setCursor(4, 1);
    estadoregistro = 3;
```

```
}

//INTRODUCIENDO LA CONFIRMACION DE CLAVE
if (estadoregistro == 3) {
  char key = keypad.getKey();
  if (key) {
    if (key == 'E') {
      if (nuevaPsw == repeatPsw) {
        estadoregistro = 4;
      } else {
        estadoregistro = 8;
        repeatPsw = "";
      }
    } else {
      lcd.print(key);
      repeatPsw = repeatPsw  + key;
    }
  }
}

//MOSTRANDO MENSAJE: CLAVE GERENTE
if (estadoregistro == 4) {

  lcd.clear();
  lcd.setCursor(1, 0);
  lcd.print("CLAVE GERENTE:");
  lcd.setCursor(7, 1);
  estadoregistro = 9;
}

//INTRODUCIENDO LA CLAVE DEL GERENTE
if (estadoregistro == 9) {

  char key = keypad.getKey();
  if (key) {
    if (key == 'E') {
      estadoregistro = 2;
      if (adminPsw == "0106") {
        adminPsw = "";
        estadoregistro = 5;
        lcd.clear();
      } else {
        estadoregistro = 6;
        adminPsw = "";
      }
```

```
    } else {
      lcd.print(key);
      adminPsw = adminPsw  + key;
    }
  }
}



//REGISTRO EXITOSO
if (estadoregistro == 5) {
  lcd.setCursor(0, 0);
  lcd.print("REGISTRO EXITOSO");
  lcd.setCursor(1, 1);

  int randomNumber;
  randomNumber = random(2300, 6982);
  String nuevoID = "#" + String(randomNumber) + "*";
  lcd.print("SU ID: " + nuevoID);

  //----------------GUARDANDO EN MEMORIA---------------------
  char IDN[10];
  char PSWN[10];
  nuevoID.toCharArray(IDN, 10);
  nuevaPsw.toCharArray(PSWN, 10);
  EMPLEADO nuevo;
  strncpy(nuevo.ID, IDN, sizeof(nuevo.ID));
  strncpy(nuevo.PSW, PSWN, sizeof(nuevo.PSW));
  int registrados = 0;
  EEPROM.get(0, registrados);
  int puntero = 2 + (registrados * 20);
  EEPROM.put(puntero, nuevo);
  registrados = registrados + 1;
  EEPROM.put(0, registrados);
  //-------------------------------------------------------------
  nuevaPsw = "";
  adminPsw = "";
  repeatPsw = "";

  delay(4000);

  estadoregistro = 0;
  estadoRegistro = false;
  estadoID = true;
```

```
    lcd.clear();
    lcd.setCursor(1, 0);
    lcd.print("IDENTIFICADOR:");
    lcd.setCursor(6, 1);
  }

  //ERROR: CLAVE GERENTE  INCORRECTA
  if (estadoregistro == 6) {
    lcd.clear();
    lcd.setCursor(3, 0);
    lcd.print("INCORRECTA");
    lcd.setCursor(6, 1);
    lcd.print("XXX");
    delay(3000);
    estadoregistro = 4;
  }

  //ERROR: CONTRASEÑA SUPERA 8 DIGITOS
  if (estadoregistro == 7) {

    lcd.setCursor(4, 0);
    lcd.print("EXCEDE 8");
    lcd.setCursor(5, 1);
    lcd.print("DIGITOS");
    delay(3000);
    estadoregistro = 0;
    nuevaPsw = "";
  }

  //ERROR: CONTRASEÑA NO COINCIDE
  if (estadoregistro == 8) {
    lcd.clear();
    lcd.setCursor(7, 0);
    lcd.print("NO");
    lcd.setCursor(4, 1);
    lcd.print("COINCIDEN");
    delay(3000);
    estadoregistro = 2;
    repeatPsw = "";
  }
}

boolean ValidacionDatos(String IDE, String Password) {
  int cant = 0;
  EEPROM.get(0, cant);
```

```
  for (int i = 0; i < cant; i++) {
    EMPLEADO emp;
    EEPROM.get(2 + (i * 20), emp);
    if (String(emp.ID) == IDE && String(emp.PSW) == Password) {
      //Serial.println("Encontrado");
      return true;
    }
  }
  return false;
}


void MostrarTodo() {
  int cant = 0;
  EEPROM.get(0, cant);
  for (int i = 0; i < cant; i++) {
    EMPLEADO emp;
    EEPROM.get(2 + (i * 20), emp);
    Serial.println("#" + i + 1);
    Serial.print("ID: ");
    Serial.println(emp.ID);
    Serial.print("PASSWORD: ");
    Serial.println(emp.PSW);
    Serial.println("");
    delay(100);
  }
}

void SPKBLOQUEO() {
  tone(10, 200 , 80);
  delay(500);
  noTone(10);
  delay(500);
}

void SPKBIENVENIDO() {
  tone(10, 600 , 80);
  delay(500);
  noTone(10);
  delay(500);
}

void msg_bienvenida() {
  lcd.setCursor(1, 0);
  lcd.write(byte(1));
```

```
  lcd.setCursor(3, 0);
  lcd.print("BIENVENIDO");
  lcd.setCursor(14, 0);
  lcd.write(byte(1));
  delay(5000);
}
```