PROYECTO ARDUINO ASCENSOR

María Ortega Monge 54777

Laura Villajos Mateo 54904

Índice:

Descripción del trabajo:

Nuestro proyecto para la asignatura de informática es un ascensor con una placa Arduino y controlada por un programa que hemos elaborado en C.

Las funciones que realiza nuestro trabajo son:

- Como es evidente, la función principal es subir y bajar la plataforma del ascensor.
- Contamos con un semáforo de tres colores (rojo, azul y verde) mediante unos LEDs incorporados en la Breadboard.
- Además, emite sonido con el movimiento.
- Por último, tenemos una pantalla LCD.

Para realizar nuestro proyecto hemos utilizado distintos instrumentos y materiales:

-Para lo físico:

- Cartón: para simular el ascensor y el decorado.
- Cuerdas
- Poleas
- Bridas

-Para el Hardware:

- Placa Arduino Uno
- Breadboard
- LEDs
- Cables
- Dos pulsadores
- Resistencias
- Batería
- Servomotor
- Pantalla LCD
- Potenciómetro para la pantalla

FUNCIONAMIENTO DE LAS PARTES:

El mecanismo utilizado para subir y bajar parte de un servomotor que activamos mediante las funciones “subir” y “bajar” del programa que explicaremos más adelante. Además también tenemos otra función llamada “para” que aparece al inicio del programa y después de realizar las funciones de subir y bajar. El servomotor está conectado a la placa Arduino y a la Breadboard. Así como los dos pulsadores con los que contamos, uno para subir y otro para bajar.

Para el “semáforo” hemos utilizado tres LEDs: Cuando el ascensor está arriba, rojo, es decir no se puede pasar. Cuando está en movimiento tanto para subir como para bajar el LED azul se activará y significa “espere”. Y por último el color verde se encenderá cuando el ascensor esté abajo dando permiso para acceder a él. Los tres LEDs se encuentran conectados a la Breadboard con unas resistencias de 200 Ohmios y a la placa Arduino.

El sonido entra en escena con el movimiento. Lo hemos puesto a modo de semáforo para personas invidentes. Conseguimos el sonido mediante un buzzer conectado en la Breadboard.

En la pantalla hemos escrito el mensaje “Trabajo ascensor” con ayuda del programa en Arduino y la función setup.

Funciones y bibliotecas añadidas al programa:

Para elaborar el programa y usar arduino hemos utilizado funciones para tratar las entradas y salidas principales.

-DigitalWrite: Permite escribir valores digitales en un pin de salida de una tarjeta Arduino. Para ello necesita que el pin haya sido declarado previamente. Necesita dos parámetros de entrada, el número PIN de la tarjeta Arduino que le corresponda (0-13). Y el estado, o LOW o HIGH, son contrarios, cuando LOW indique 0 HIGH indicará 1. Esta función no tiene parámetros de salida y es usada para motores de dc, motores a paso, LEDs, pantallas LCD, servomotores, buzzer, etc.

En nuestro caso, la hemos usado para el servomotor, LEDs y buzzer.

-PinMode(): Permite configurar cada pin de manera individual tanto de entrada como de salida. Se utiliza (#PIN, MODO)

- #PIN: número de pin que queremos configurar.
- MODO:

  -INPUT: configura el pin como entrada.

  -OUTPUT: configura el pin como salida.

  -INPUT_PULLUP: se le agrega al pin una resistencia interna y se configura como entrada.

-DigitalRead(): Lee el valor del pin correspondiente como HIGH o LOW. Dentro del paréntesis va el pin.

-Delay: retardos de tiempo en milisegundos. Es una función que permite hacer que el procesador espere y que no haga nada hasta la ejecución de la siguiente instrucción durante un retardo de tiempo definido.

-ServoAttach(): Dentro del paréntesis incluimos el número pin que le asignamos al servomotor.

-Tone/NoTone: Sirve para el sonido, debemos describir la frecuencia a la que lo queremos y el numero pin del buzzer.

Para que algunas de estas funciones pudieran ejecutarse hemos tenido que añadir nuevas librerías como:

#include<servo.h>: Para servoattach() y para la declaración del servomotor.

#include <LiquidCrystal.h>: Sirve para las funciones de la pantalla LCD.

Y para conectar Arduino a C hemos utilizado las funciones Serial.print() y Serial.begin().

Programa definitivo:

```
#include <LiquidCrystal.h>
#include <Servo.h>
Servo myservo;
const int pinBuzzer = 12;
int RS = 4, t E = 2, D4 = 1, D5 = 10, D6 = 13,
D7 = 8, VO = 3;
LiquidCrystal lcd (RS, E, D4, D5, D6, D7);
const int  inPin1=5;//
const int  inPin2=7;//
const int  ROJOPin=9;
const int  VERDEPin=10;
int pa=LOW;
```

Declaración de las bibliotecas del servo y la pantalla y variables de tipo int.

```
void baja ()// variamos los valores de giro para que giren las poleas
{ tone(pinBuzzer, 650);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
 for(int i =0;i<95;i=i+2)
 {myservo.write(i);
  delay(100);
 }
 delay(1000);
 digitalWrite(8,LOW);
 noTone(pinBuzzer);
}
```

Función baja: Empieza el sonido una vez empieza a moverse con una frecuencia de 650 Hz. El servomotor se mueve cada dos grados gracias al bucle for. Y añadimos los delays para retardar la duración de una función como el sonido o la duración de la luz de los LEDs.

```
void sube()// variamos los valores de giro para
 que giren al lado contrario
{
   tone(pinBuzzer, 650);
digitalWrite(8,HIGH);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
 for(int i=90;i>=0;i=i-2)
 {
  myservo.write(i);
  delay(100);
 }
 delay(1000);
 digitalWrite(8,LOW);
 noTone(pinBuzzer);
}
```

Función sube : al igual que la función baja empieza el sonido una vez empieza a moverse con una frecuencia de 650 Hz. El servomotor se mueve cada dos grados gracias al bucle for en sentido contrario de la función baja. Y añadimos los delays para retardar la duración de una función como el sonido o la duración de la luz de los LEDs.

```
void para () //
la utilizaremos al inicio y cuando esté bajando y
se pulse el botón subir.
{digitalWrite(inPin1,LOW);//ponemos los pines  6 y 7 en
balor bajo
digitalWrite(inPin2,LOW);
}
```

Función para: función para que el motor no se mueva. Los dos InPin están en LOW porque no hemos pulsado los botones.

```
void setup() {
  // put your setup code here, to run once:
  analogWrite(VO, 50);
    lcd.begin(16,2);
    lcd.setCursor (0,0);
    lcd.print("trabajo");
    lcd.setCursor (0,1);
    lcd.print("ASCENSOR");
  Serial.begin(9600);
  pinMode(pinBuzzer,OUTPUT);
myservo.attach(6);
pinMode(inPin1,INPUT);
pinMode(inPin2,INPUT);
  pinMode(8,OUTPUT); //Rojo
  pinMode(9,OUTPUT); //Azul
  pinMode(10,OUTPUT); //Verde
}
```

Declaración de los pines de entrada y salida.

```
void loop() {
para();
pa=LOW;
while(pa==LOW)
{pa = digitalRead(inPin1);
digitalWrite(9,HIGH);
digitalWrite(10,LOW);
}
pa=LOW;
 sube();
 para();
pa=LOW;
while(pa==LOW)
{pa = digitalRead(inPin2);
digitalWrite(10,HIGH);
digitalWrite(9,LOW);
}
pa=LOW;//reinicializamos pa
 baja();
 }
```

Cuerpo del programa: Empieza con el contador pa (pulsadores) en LOW y mientras no pulsemos uno sigue en el bucle while. Si el pa cambia de valor a HIGH comienzan las funciones sube y baja dependiendo de que botón utilices.

Como lo hemos conectado al C:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "SerialPort.h"
#include "SerialPort.C"
#define MAX_DATA_LENGTH 255
void autoConnect(SerialPort *arduino,char*);
int main(void)
{//Arduino SerialPort object
SerialPort *arduino;
char* portName = "\\\\.\\COM3";
char incomingData[MAX_DATA_LENGTH];
arduino = (SerialPort *)malloc(sizeof(SerialPort));
Crear_Conexion(arduino,portName);
autoConnect(arduino,incomingData);
return 0;
}
void autoConnect(SerialPort *arduino,char *incomingData)
{char sendData = 0;
int readResult;
// Espera la conexión con Arduino
while (!isConnected(arduino))
{Sleep(100);
Crear_Conexion(arduino,arduino->portName);
}
//Comprueba si arduino está connectado
if (isConnected(arduino))
{printf ("Conectado con Arduino en el puerto %s\n",arduino->portName);
}
```

Incluimos las bibliotecas necesarias. Hemos descargado la biblioteca SerialPort de Moodle y la hemos descomprimido en la misma carpeta en la que hemos guardado el programa porque si no, el programa no encuentra los datos de la biblioteca y no funciona.

Funciones prototipo.

Puerto Serie en el que está Arduino.

Se crea la estructura de datos del puerto serie.

Apertura del puerto serie.

Espera la conexión con Arduino.

Comprueba que Arduino esté conectado.

```
printf ("0 - OFF, 1 - ON, 9 - SALIR\n");
while (isConnected(arduino) && sendData!='9')
{sendData = getch();
writeSerialPort(arduino,&sendData, sizeof(char));
readResult=readSerialPort(arduino,incomingData,MAX_DATA_LENGTH);
if (readResult!=0)
printf ("%s",incomingData);
sleep(10);
}
if (!isConnected(arduino))
printf ("Se ha perdido la conexión con Arduino\n");
}
```

Bucle de la aplicación.

BIBLIOGRAFÍA:

- Tutoriales de youtube para aprender a conectar los elementos a la placa: https://www.youtube.com/user/codigofacilito
- Para encontrar las funciones necesarias: https://hetpro-store.com/TUTORIALES/arduino-digitalwrite/
- https://www.luisllamas.es/reproducir-sonidos-arduino-buzzer-pasivo-altavoz/
- Moodle.