

**INEBRIATOR**

ARDUINO + JAVA



23 de junio de 2016

Universidad Privada del Estado de México

Ixtapaluca, México

ISLAS CUERO DANIEL

INGENIERIA EN SISTEMAS

ING ESLAVA MONTES GABRIEL

ING TREJO ESPINOZA GERARDO

INDICE

[ARDUINO + JAVA 4](#_Toc454524664)

[INTRODUCCION 4](#_Toc454524665)

[THE INEBRIATOR 5](#_Toc454524666)

[HARDWARE LIBRE Y SOFTWARE LIBRE 5](#_Toc454524667)

[HARDWARE LIBRE 5](#_Toc454524668)

[SOFTWARE LIBRE 5](#_Toc454524669)

[ARDUINO 5](#_Toc454524670)

[ELECTRONICA 6](#_Toc454524671)

[VOLTAJE 6](#_Toc454524672)

[CORRIENTE 6](#_Toc454524673)

[RESISTENCIA 7](#_Toc454524674)

[LEY DE OHM 7](#_Toc454524675)

[SISTEMAS ELECTRONICOS 7](#_Toc454524676)

[SALIDAS 7](#_Toc454524677)

[SEÑALES ELECTRONICAS 8](#_Toc454524678)

[COMUNICACIÓN SERIAL 8](#_Toc454524679)

[PROTOBOARD 8](#_Toc454524680)

[RESISTENCIA 8](#_Toc454524681)

[DIODO 8](#_Toc454524682)

[TRANSISTOR 9](#_Toc454524683)

[CONDENSADOR 9](#_Toc454524684)

[MOTOR DC 9](#_Toc454524685)

[PLACA ARDUINO UNO 9](#_Toc454524686)

[PARTES DE ARDUINO UNO 10](#_Toc454524687)

[CARACTERISTICAS DE LA PLACA 10](#_Toc454524688)

[Entradas y salidas digitales/analógicas 11](#_Toc454524689)

[Terminales Digitales 11](#_Toc454524690)

[Pines Analógicos 11](#_Toc454524691)

[Pines de alimentación 12](#_Toc454524692)

[PROGRAMACIÓN 12](#_Toc454524693)

[LENGUAJE DE PROGRAMACION 13](#_Toc454524694)

[PROGRAMACION EN ARDUINO 13](#_Toc454524695)

[INTERFAZ GRAFICA EN JAVA 14](#_Toc454524696)

[CODIGO: 14](#_Toc454524697)

[CONTROLADOR ARDUINO 28](#_Toc454524698)

# ARDUINO + JAVA

## INTRODUCCION

Conocer el funcionamiento de las cosas es algo que nos hemos preguntado desde el inicio de la historia; en nuestros días y desde la invención de la rueda y la revolución industrial nos enfrentamos a una realidad donde abundan la automatización, la domótica (automatización de las casas y edificios), la interacción de las personas con las máquinas, la electrónica, la mecánica y la programación. Casi cualquier proceso que nos podamos imaginar tiene un porcentaje de dependencia de estas máquinas. El propósito de esta guía es abordar el concepto de computación física que es la capacidad de interacción y comunicación de una máquina con los humanos, usando sensores y actuadores. Las decisiones de esto las va a tomar un microcontrolador que se encuentra ubicado en la placa Arduino. La tarjeta Arduino es el corazón del proyecto al que nombramos inebriator.

La Computación física, significa la construcción de sistemas interactivos físicos mediante el uso de software y hardware que pueden sentir y responder al mundo analógico. Si bien esta definición es suficientemente amplia para abarcar aspectos como los sistemas inteligentes de control de tráfico de automóviles o los procesos de automatización de fábricas, en un sentido más amplio, la computación física es un marco creativo para la comprensión de la relación de los seres humanos en el mundo digital. En la práctica, a menudo el término describe el arte hecho a mano, diseño de proyectos DIY o pasatiempos que utilizan sensores y microcontroladores para traducir entradas analógicas a sistemas basados en software, y/o controlar dispositivos electromecánicos como motores, servos, iluminación u otro hardware. Otras implementaciones de computación física trabajan con el reconocimiento de la voz, la cual se capta e interpretan sus ondas sonoras a través de micrófonos u otros dispositivos de detección de ondas sonoras, también la visión por computador, que aplica algoritmos a los videos detectados por algún tipo de cámara. Interfaces táctiles son también un ejemplo de la computación física. El prototipado (crear montajes rápidos con ayuda de una Protoboard y componentes básicos de electrónica) juega un papel importante en la computación física. Herramientas como Arduino y Fritzing son útiles para diseñadores, artistas, estudiantes y entusiastas porque ayudan a elaborar prototipos rápidamente.

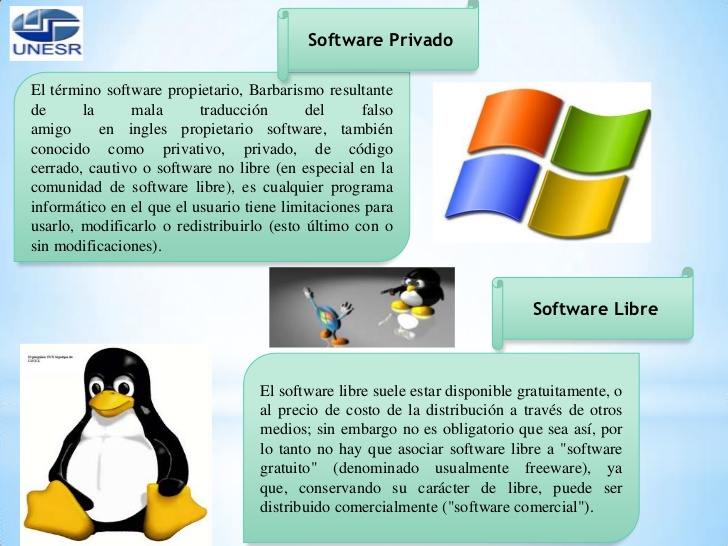
## THE INEBRIATOR

El proyecto llamado The Inebriator, trata de la automatización al momento de servir tragos, este proyecto representa la solución al problema del servicio al cliente en ciertos establecimientos, ya que lo hace con rapidez y seguridad.

Pero antes veremos algunas particularidades con la cual iniciar el proyecto.

## HARDWARE LIBRE Y SOFTWARE LIBRE

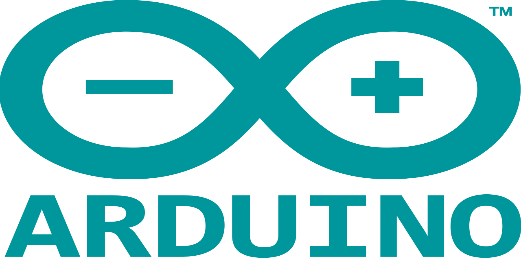
### HARDWARE LIBRE

Se llama hardware libre a los dispositivos de hardware cuyas especificaciones y diagramas esquemáticos son de acceso público, ya sea bajo algún tipo de pago o de forma gratuita. La filosofía del software libre (las ideas sobre la libertad del conocimiento) es aplicable a la del hardware libre. Se debe recordar en todo momento que libre no es sinónimo de gratis. El hardware libre forma parte de la cultura libre. Dado que el hardware tiene asociados a él costos variables directos, ninguna definición de software libre se puede aplicar directamente sin modificación. En cambio, el término hardware libre se ha usado principalmente para reflejar el uso del software libre con el hardware y el lanzamiento libre de la información con respecto al hardware, a menudo incluyendo el lanzamiento de los diagramas esquemáticos, diseños y montajes.

### SOFTWARE LIBRE

El software libre (en inglés free software, aunque esta denominación también se confunde a veces con "gratis" por la ambigüedad del término "free" en el idioma inglés, por lo que también se usa "libre software" y "lógica libre") es la denominación del software que respeta la libertad de los usuarios sobre su producto adquirido y, por tanto, una vez obtenido puede ser usado, copiado, estudiado, modificado, y redistribuido libremente. Según la Free Software Foundation, el software libre se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, modificar el software y distribuirlo modificado.

## ARDUINO

Arduino es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware libre, flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquier interesado en crear entornos u objetos interactivos. Arduino puede tomar información del entorno a través de sus pines de entrada, para esto toda una gama de sensores puede ser usada y puede afectar aquello que le rodea controlando luces, motores y otros actuadores. El microcontrolador en la placa Arduino se programa mediante el lenguaje de programación Arduino (basado en Wiring) y el entorno de desarrollo Arduino (basado en Processing). Los proyectos hechos con Arduino pueden ejecutarse sin necesidad de conectarlo a un ordenador, si bien tienen la posibilidad de hacerlo y comunicar con diferentes tipos de software (p.ej. Flash, Processing, MaxMSP). Las placas pueden ser hechas a mano o comprarse montadas de fábrica; el software puede ser descargado de forma gratuita. Los ficheros de diseño de referencia (CAD) están disponibles bajo una licencia abierta, así pues, eres libre de adaptarlos a tus necesidades.

## ELECTRONICA

Estudia y emplea sistemas cuyo funcionamiento se basa en la conducción y el control del flujo de los electrones u otras partículas cargadas eléctricamente. El diseño y la gran construcción de circuitos electrónicos para resolver problemas prácticos forman parte de la electrónica y de los campos de la ingeniería electrónica, electromecánica y la informática en el diseño de software para su control. La electrónica desarrolla en la actualidad una gran variedad de tareas. Los principales usos de los circuitos electrónicos son el control, el procesado, la distribución de información, la conversión y la distribución de la energía eléctrica. Estos dos usos implican la creación o la detección de campos electromagnéticos y corrientes eléctricas.

## VOLTAJE

Una magnitud física que impulsa a los electrones a lo largo de un conductor (por ejemplo, un cable) en un circuito eléctrico cerrado, provocando el flujo de una corriente eléctrica. Su unidad es el Voltio(V).

1. Voltaje DC Es el flujo continuo de electrones a través de un conductor entre dos puntos de distinto potencial. En la corriente continua las cargas eléctricas circulan siempre en la misma dirección, es continua la corriente mantiene siempre la misma polaridad. En la norma sistemática europea el color negro corresponde al negativo y el rojo al positivo o sencillamente se simboliza para el positivo con VCC, +, VSS y para el negativo con 0V, -, GND. Muchos aparatos necesitan corriente continua para funcionar, sobre todos los que llevan electrónica (equipos audiovisuales, computadores, etc.), para ello se utilizan fuentes de alimentación. Lo puedes encontrar en la batería, pilas, salida de los cargadores de computador.
2. Voltaje AC Es la corriente eléctrica en la que la magnitud y dirección varían cíclicamente. La forma de onda de la corriente alterna más comúnmente utilizada es la de una onda seno. El voltaje AC es el que llega a la toma de electricidad de los hogares y a las empresas, es muy común encontrarla en las tomas de corriente donde se conectan nuestros electrodomésticos. Sin embargo, las señales de audio y de radio transmitidas por los cables eléctricos son también ejemplos de corriente alterna. En estos usos, el fin más importante suele ser la transmisión y recuperación de la información codificada (o modulada) sobre la señal de la AC.

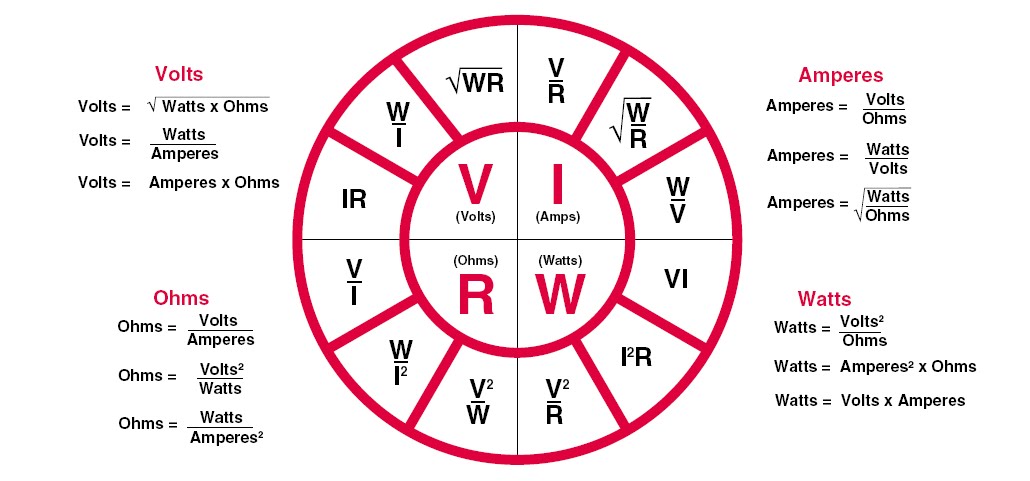
## CORRIENTE

Es el flujo de electrones a través de un conductor o semiconductor en un sentido. La unidad de medida de ésta es el amperio (A). Una corriente eléctrica, puesto que se trata de un movimiento de cargas, produce un campo magnético, un fenómeno que puede aprovecharse en el electroimán, este es el principio de funcionamiento de un motor. El instrumento usado para medir la intensidad de la corriente eléctrica es el galvanómetro que, calibrado en amperios, se llama amperímetro, colocado en serie con el conductor cuya intensidad se desea medir.

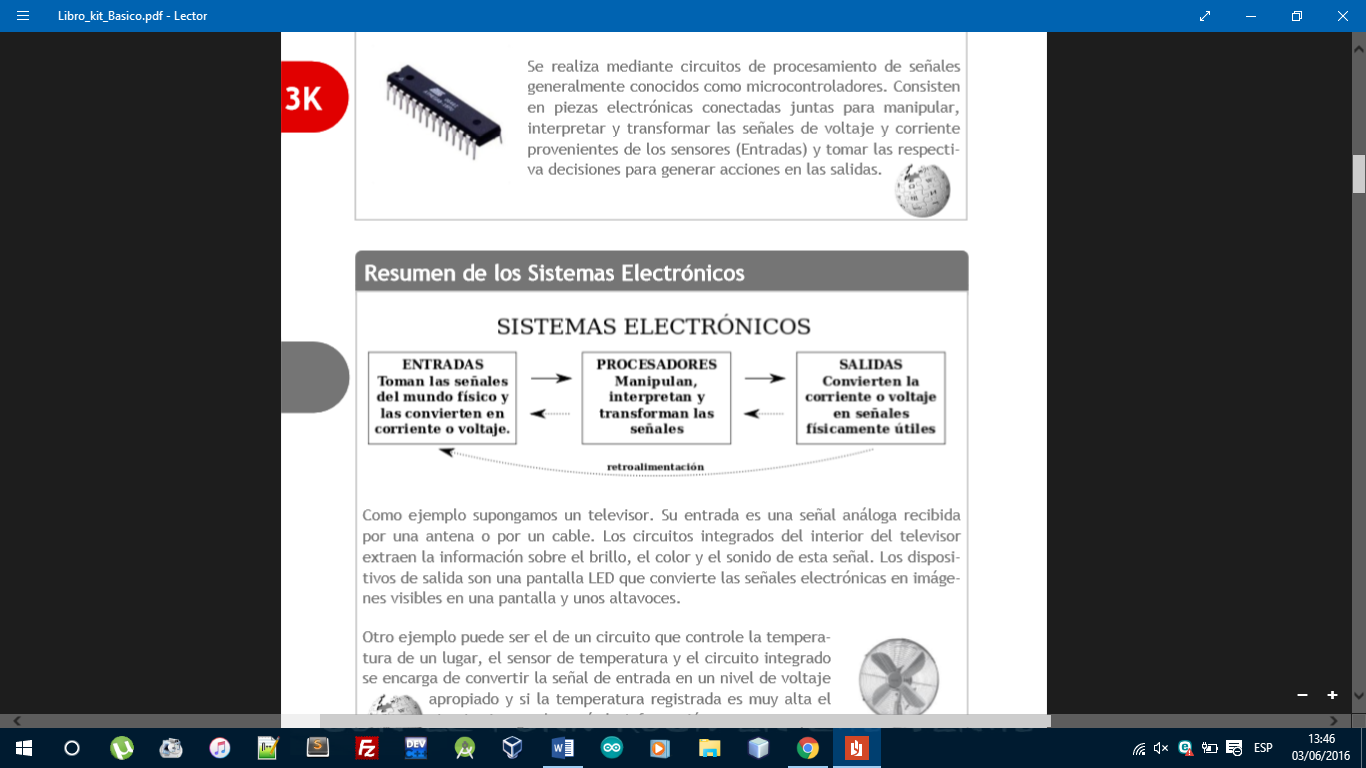
## RESISTENCIA

Es la propiedad física mediante la cual todos los materiales tienden a oponerse al flujo de la corriente. La unidad de este parámetro es el Ohmio (Ω). Puedes encontrar resistencias en los calefactores eléctricos, tarjetas electrónicas, estufas son muy útiles para limitar el paso de la corriente.

## LEY DE OHM

La ley dice que la corriente (I) que circula por un conductor eléctrico es directamente proporcional al voltaje (V) e inversamente proporcional a la resistencia (R). La pirámide de la derecha es muy útil para conocer la fórmula a la que es igual la variable que tapes con el dedo, por ejemplo: Tapa con tu dedo la V (voltaje), entonces voltaje va a ser igual a I (corriente) por R (resistencia), una más, tapa I (Corriente), I va ser igual a V divido R.

## SISTEMAS ELECTRONICOS

Un sistema electrónico es un conjunto de circuitos que interactúan entre sí para obtener un resultado. Una forma de entender los sistemas electrónicos consiste en dividirlos en entradas, salidas y procesamiento de señal.

## SALIDAS

Son actuadores u otros dispositivos (también transductores) que convierten las señales de corriente o voltaje en señales físicamente útiles como movimiento, luz, sonido, fuerza, rotación entre otros. Por ejemplo: un motor que gire, un LED o sistema de luces que se encienda automáticamente cuando esté oscureciendo, un buzzer que genere diversos tonos.

## SEÑALES ELECTRONICAS

Las entradas y salidas de un sistema electrónico serán consideradas como las señales variables. En electrónica se trabaja con variables que se toman en forma de voltaje o corriente, éstas se pueden denominar comúnmente señales. Las señales primordialmente pueden ser de dos tipos descritos a continuación: Digital o análoga.

1. Variable digital También llamadas variables discretas. Se caracterizan por tener dos estados diferenciados y por lo tanto se pueden llamar binarias. Siendo estas variables más fáciles de tratar (en lógica serían los valores Verdadero (V) y Falso (F) o podrían ser 1 ó 0 respectivamente). Un ejemplo de una señal digital es el interruptor del timbre de tu casa, porque este interruptor tiene dos estados pulsado y sin pulsar
2. Variable análoga Son aquellas que pueden tomar un número infinito de valores comprendidos entre dos límites. La mayoría de los fenómenos de la vida real son señales de este tipo. (sonido, temperatura, voz, video, etc.) Un ejemplo de sistema electrónico analógico es un parlante, que se emplea para amplificar el sonido de forma que éste sea oído por una gran audiencia. Las ondas de sonido que son analógicas en su origen, son capturadas por un micrófono y convertidas en una pequeña variación analógica de tensión denominada señal de audio.

## COMUNICACIÓN SERIAL

Es una interfaz de comunicaciones de datos digitales, frecuentemente utilizado por computadores y periféricos, donde la información es transmitida bit a bit enviando un solo bit a la vez. Uno de sus usos es monitorear a través de la pantalla del computador el estado del periférico conectado, por ejemplo, al pulsar la letra A en el teclado se debe encender un LED conectado de manera remota al computador.

## PROTOBOARD

Es una placa reutilizable usada para construir prototipos de circuitos electrónicos sin soldadura. Compuestas por bloques de plástico perforados y numerosas láminas delgadas de una aleación de cobre, estaño y fósforo.

## RESISTENCIA

Es un material formado por carbón y otros elementos resistivos para disminuir la corriente que pasa. Se opone al paso de la corriente. La corriente máxima en un resistor viene condicionada por la máxima potencia que puede disipar su cuerpo. Esta potencia se puede identificar visualmente a partir del diámetro sin que sea necesaria otra indicación. Los valores más comunes son 0,25 W, 0,5 W y 1 W.

El valor de la resistencia eléctrica se obtiene leyendo las cifras como un número de una, dos o tres cifras; se multiplica por el multiplicador y se obtiene el resultado en Ohmios (Ω).

## DIODO

Un diodo es un componente electrónico de dos terminales que permite la circulación de la corriente eléctrica a través de él en un solo sentido. Tiene dos partes: el cátodo y el ánodo.

## TRANSISTOR

El transistor es un dispositivo electrónico semiconductor que cumple funciones de amplificador, oscilador, conmutador o rectificador. Tiene tres partes: la base (B), el emisor (E) y colector (C).

Actualmente se encuentran prácticamente en todos los aparatos domésticos de uso diario: radios, televisores, grabadoras, reproductores de audio y video, hornos de microondas, lavadoras, automóviles, equipos de refrigeración, alarmas, relojes de cuarzo, ordenadores, calculadoras, impresoras, lámparas fluorescentes, equipos de rayos X, tomógrafos, ecógrafos, reproductores mp3, teléfonos celulares, etc.

## CONDENSADOR

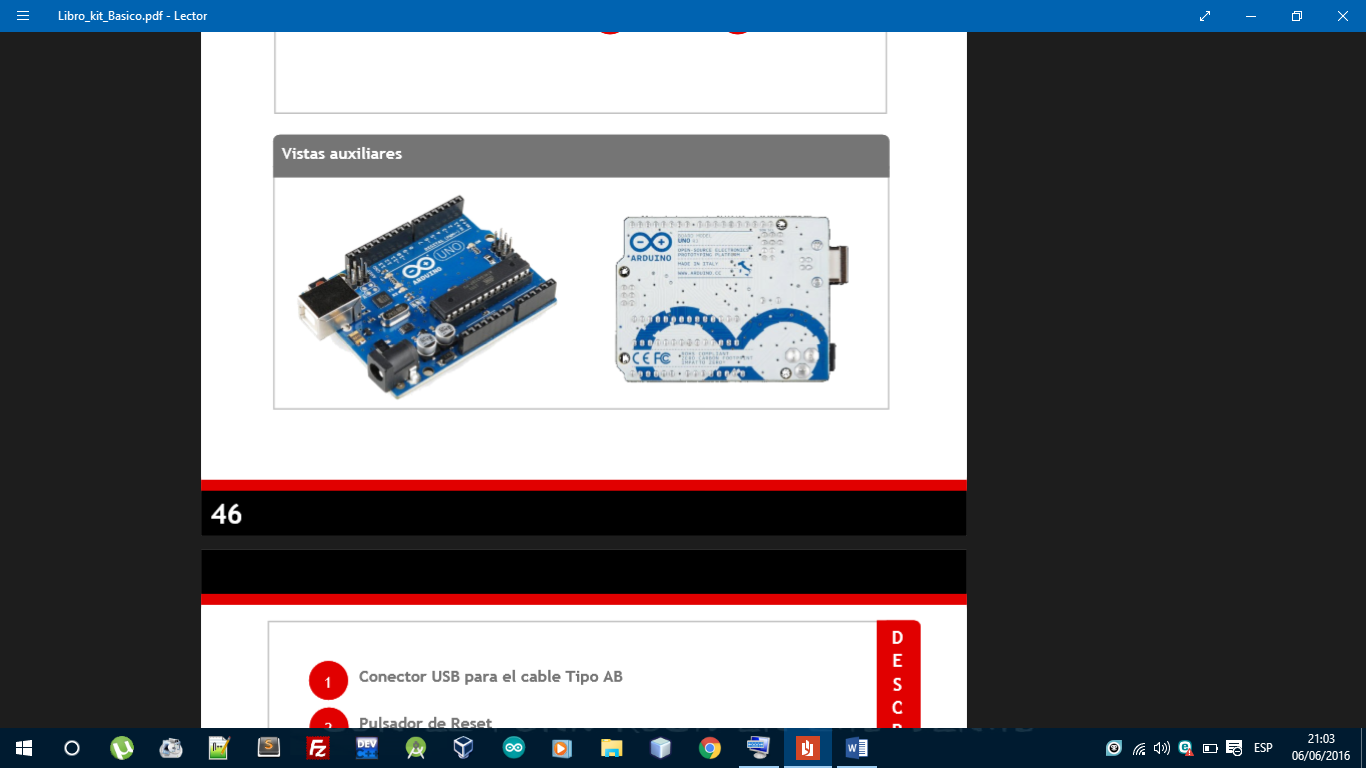
Un condensador o capacitor es un dispositivo utilizado en electrónica, capaz de almacenar energía sustentando un campo eléctrico. Está formado por un par de superficies conductoras, generalmente en forma de láminas o placas, en situación de influencia total separadas por un material dieléctrico o por el vacío. Las placas, sometidas a una diferencia de potencial, adquieren una determinada carga eléctrica, positiva en una de ellas y negativa en la otra.

## MOTOR DC

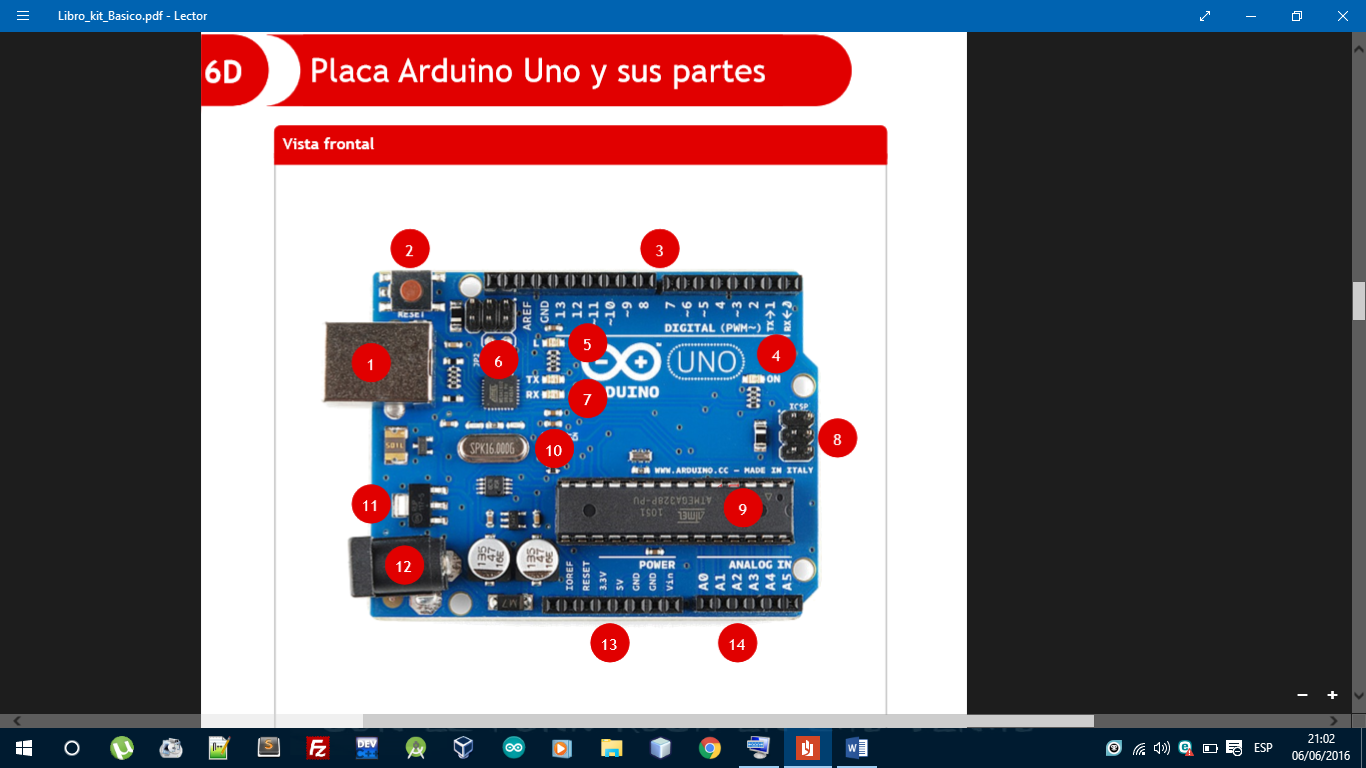
El motor de corriente continua (DC) es una máquina que convierte la energía eléctrica en mecánica, provocando un movimiento rotatorio. Esta máquina de corriente continua es una de las más versátiles. Su fácil control de posición, paro y velocidad la han convertido en una de las mejores opciones en aplicaciones de control y automatización de procesos. Por ejemplo, los puedes encontrar en la tracción de los carros de juguetes de pilas o en las llantas de los robots.

## PLACA ARDUINO UNO

Es una placa electrónica basada en el microcontrolador ATmega328. Cuenta con 14 entradas/salidas digitales, de las cuales 6 se pueden utilizar como salidas PWM (Modulación por ancho de pulsos) y otras 6 son entradas analógicas. Además, incluye un resonador cerámico de 16 MHz, un conector USB, un conector de alimentación, una cabecera ICSP y un botón de reseteado. La placa incluye todo lo necesario para que el microcontrolador haga su trabajo, basta conectarla a un ordenador con un cable USB o a la corriente eléctrica a través de un transformador.



## PARTES DE ARDUINO UNO



1. Conector USB para el cable Tipo AB
2. Pulsador de Reset
3. Pines de E/S digitales y PWM
4. LED verde de placa encendida
5. LED naranja conectado al pin13
6. ATmega 16U2 encargado de la comunicación con el PC
7. LED TX (Transmisor) y RX (Receptor) de la comunicación serial
8. Puerto ICSP para programación serial
9. Microcontrolador ATmega 328, cerebro del Arduino
10. Cristal de cuarzo de 16Mhz
11. Regulador de voltaje
12. Conector hembra 2.1mm con centro positivo
13. Pines de voltaje y tierra
14. Entradas análogas

## CARACTERISTICAS DE LA PLACA

1. Microcontrolador: ATmega328
2. Voltage: 5V
3. Voltage entrada (recomendado): 7-12V
4. Voltage entrada (limites): 6-20V
5. Digital I/O Pins: 14 (de los cuales 6 son salida PWM)
6. Entradas Analogicas: 6
7. DC Current per I/O Pin: 40 mA
8. DC Current parar 3.3V Pin: 50 mA
9. Flash Memory: 32 KB (ATmega328) de los cuales 0.5 KB son utilizados para el arranque
10. SRAM: 2 KB (ATmega328)
11. EEPROM: 1 KB (ATmega328)
12. Clock Speed: 16 MHz

### Entradas y salidas digitales/analógicas

Un sistema electrónico es cualquier disposición de componentes electrónicos con un conjunto definido de entradas y salidas. Una placa Arduino, por tanto, puede pensarse de forma simplificada como un sistema que acepta información en forma de señal de entrada, desarrolla ciertas operaciones sobre ésta y luego produce señales de salida.

Justamente, una de las opciones que hacen más potente a Arduino son sus entradas/salidas digitales. ¿Entonces por qué hablamos de analógicas?

En los sistemas electrónicos, una magnitud física variable se representa generalmente mediante una señal eléctrica que varía de manera tal que describe esa magnitud. Por lo general, se hace referencia a las señales continuas como señales analógicas, mientras que asociamos las señales discretas a señales digitales: el ejemplo más claro es el de las señales binarias, donde la señal sólo pueden tomar dos niveles, 0 o 1.

Arduino incorpora terminales digitales (señales discretas) pero de tal forma que tenemos un gran abanico de valores con los que trabajar (por ejemplo, 255 valores de luz en un fotosensor, siendo 0 ausencia de luz y 255 el máximo valor lumínico).

### Terminales Digitales

Las terminales digitales de una placa Arduino pueden ser utilizadas para entradas o salidas de propósito general a través de los comandos de programación pinMode(), digitalRead(), y digitalWrite(). Cada terminal tiene una resistencia pull-up que puede activarse o desactivarse utilizando DigitalWrite() (con un valor de HIGH o LOW, respectivamente) cuando el pin esta configurado como entrada. La corriente máxima por salida es 40 mA.

1. Serial: 0 (RX) y 1 (TX). Utilizado para recibir (RX) y transmitir (TX) datos serie TTL. En el Arduino Diacemila, estas terminales están conectadas a las correspondientes patas del circuito integrado conversor FTDI USB a TTL serie. En el Arduino BT, están conectados al las terminales correspondientes del modulo Bluetooth WT11. En el Arduino Mini y el Arduino LilyPad, están destinados para el uso de un módulo serie TTL externo (por ejemplo el adaptador Mini-USB).
2. Interruptores externos: 2 y 3. Estas terminales pueden ser configuradas para disparar una interrupción con un valor bajo, un pulso de subida o bajada, o un cambio de valor. Mira la función attachInterrupt() para mas detalles.
3. PWM: 3, 5, 6, 9, 10, y 11. Proporcionan salidas PWM de 8 bit con la función analogWrite(). En placas con ATmega8, las salidas PWM solo están disponibles en los pines 9, 10, y 11.
4. Reset BT: 7. (solo en Arduino BT) Conectado a la línea de reset del módulo bluetooth.
5. SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Estas terminales soportan comunicación SPI. Aunque esta funcionalidad esta proporcionada por el hardware, no está incluida actualmente el el lenguaje Arduino.
6. LED: 13. En el Diacemila y el LilyPad hay un led en placa conectado al pin digital 13. cuando el pin tiene valor HIGH, el LED está encendido, cuando el pin está en LOW, está apagado

### Pines Analógicos

1. Los pines de entrada analógicos soportan conversiones analógico-digital (ADC) de 10 bit utilizando la función analogRead(). Las entradas analógicas pueden ser también usadas como pines digitales: entrada analógica 0 como pin digital 14 hasta la entrada analógica 5 como pin digital 19. Las entradas analógicas 6 y 7 (presentes en el Mini y el BT) no pueden ser utilizadas como pines digitales.
2. I2C: 4 (SDA) y 5 (SCL). Soportan comunicaciones I2C (TWI) utilizando la librería Wire (documentación en la página web de Wiring).

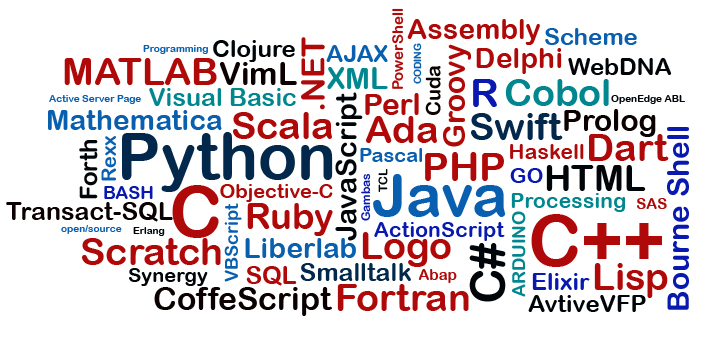
### Pines de alimentación

1. VIN (a veces marcada como "9V"). Es el voltaje de entrada a la placa Arduino cuando se está utilizando una fuente de alimentación externa (En comparación con los 5 voltios de la conexión USB o de otra fuente de alimentación regulada). Puedes proporcionar voltaje a través de este pin. Date cuenta que diferentes placas aceptan distintos rangos de voltaje de entrada, por favor, mira la documentación de tu placa. También date cuenta que el LilyPad no tiene pin VIN y acepta solo una entrada regulada.
2. 5V. La alimentación regulada utilizada para alimentar el microcontrolador y otros componentes de la placa. Esta puede venir de VIN a través de un regulador en placa o ser proporcionada por USB u otra fuente regulada de 5V.
3. 3V3. (solo en el Diacemila) Una fuente de 3.3 voltios generada por el chip FTDI de la placa.
4. GND. Pines de tierra.

Otros Pines

1. AREF. Referencia de voltaje para las entradas analógicas. Utilizada con la función analogReference().
2. Reset. (Solo en el Diacemila) Pon esta línea a LOW para resetear el microcontrolador. Utilizada típicamente para añadir un botón de reset a shields que bloquean el de la placa principal.

## PROGRAMACIÓN

La programación es un gran recurso que nos permite crear diversas secuencias de pasos lógicos que van a satisfacer nuestras necesidades y las de nuestros sistemas. Programar es todo un arte que requiere de una gran habilidad lógica y concentración por parte del programador.

Es el proceso de diseñar, escribir, probar, depurar y mantener el código fuente de programas computacionales. El código fuente es escrito en un lenguaje de programación. El propósito de la programación es crear programas que exhiban un comportamiento deseado.

El proceso de escribir código requiere frecuentemente conocimientos en varias áreas distintas, además del dominio del lenguaje a utilizar, algoritmos especializados y lógica formal. Programar involucra áreas como el análisis y diseño de la aplicación.

Para crear un programa que el computador interprete y ejecute las instrucciones escritas en él, debe usarse un Lenguaje de programación. En sus inicios los computadores interpretaban sólo instrucciones en un lenguaje específico, del más bajo nivel conocido como código máquina, siendo éste excesivamente complicado para programar. De hecho, sólo consiste en cadenas de números 1 y 0 (Sistema binario).

Para facilitar el trabajo de programación, los primeros científicos que trabajaban en el área decidieron reemplazar las instrucciones, secuencias de unos y ceros, por palabras o letras provenientes del inglés, codificándolas así y creando un lenguaje de mayor nivel, que se conoce como Assembly o lenguaje ensamblador. Por ejemplo, para sumar se usa la letra A de la palabra inglesa add (sumar). En realidad, escribir en lenguaje ensamblador es básicamente lo mismo que hacerlo en lenguaje máquina, pero las letras y palabras son bastante más fáciles de recordar y entender que secuencias de números binarios.

A medida que la complejidad de las tareas que realizaban las computadoras aumentaba, se hizo necesario disponer de un método sencillo para programar. Entonces, se crearon los lenguajes de alto nivel. Mientras que una tarea tan trivial como multiplicar dos números puede necesitar un conjunto de instrucciones en lenguaje ensamblador, en un lenguaje de alto nivel bastará con sólo una.

## LENGUAJE DE PROGRAMACION

Un lenguaje de programación es un idioma artificial diseñado para expresar operaciones que pueden ser llevadas a cabo por máquinas como los computadores. Pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana.

Está formado por un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones. Al proceso por el cual se escribe, se prueba, se depura, se compila y se mantiene el código fuente de un programa informático se le llama programación.

## PROGRAMACION EN ARDUINO

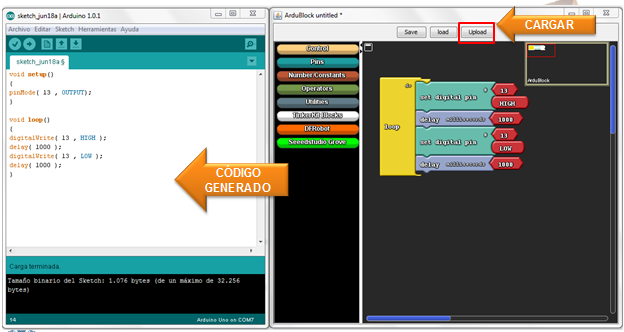
Arduino se programa en el lenguaje de alto nivel C/C++ y generalmente tiene los siguientes componentes para elaborar el algoritmo:

1. Estructuras
2. Variables
3. Operadores matemáticos, lógicos y booleanos
4. Estructuras de control (Condicionales y ciclos)
5. Funciones

Estructuras

Son dos funciones principales que debe tener todo programa en Arduino:

setup () {} Código de configuración inicial, solo se ejecuta una vez.

loop () {} Esta función se ejecuta luego del setup (), se mantiene ejecutándose hasta que se des- energice o desconecte el Arduino.

## INTERFAZ GRAFICA EN JAVA

## CODIGO:

import java.awt.Color;

import java.util.logging.Level;

import java.util.logging.Logger;

import panamahitek.Arduino.PanamaHitek\_Arduino;

import java.sql.\*;

import java.util.Calendar;

import javax.swing.JOptionPane;

import javax.swing.table.DefaultTableModel;

/\*\*

\*

\* @author SKYWALKER

\*/

public class Formulario extends javax.swing.JFrame {

PanamaHitek\_Arduino arduino = new PanamaHitek\_Arduino();

public Formulario() {

initComponents();

mostrardatos("");

mostrardatos2("");

//Color JFrame

//Color JPanel

try {

arduino.arduinoTX("COM3", 9600);

} catch (Exception ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

void mostrardatos(String valor){

DefaultTableModel modelo = new DefaultTableModel();

modelo.addColumn("NOMBRE");

modelo.addColumn("LICOR");

modelo.addColumn("JUGO");

tbproduc.setModel(modelo);

String sql="";

if(valor.equals(""))

{

sql="SELECT \* FROM bebidas";

}

else{

sql="SELECT \* FROM bebidas WHERE nom ='"+valor+"'";

}

String []datos = new String [3];

try {

Statement st = cn.createStatement();

ResultSet rs = st.executeQuery(sql);

while(rs.next()){

datos[0] = rs.getString(1);

datos[1] = rs.getString(2);

datos[2] = rs.getString(3);

modelo.addRow(datos);

}

tbproduc.setModel(modelo);

} catch (SQLException ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

void mostrardatos2(String valor){

DefaultTableModel modelo = new DefaultTableModel();

modelo.addColumn("NOMBRE");

modelo.addColumn("FECHA");

modelo.addColumn("HORA");

er.setModel(modelo);

String sql="";

if(valor.equals(""))

{

sql="SELECT \* FROM he";

}

else{

sql="SELECT \* FROM he WHERE nom ='"+valor+"'";

}

String []datos = new String [3];

try {

Statement st = cn.createStatement();

ResultSet rs = st.executeQuery(sql);

while(rs.next()){

datos[0] = rs.getString(1);

datos[1] = rs.getString(2);

datos[2] = rs.getString(3);

modelo.addRow(datos);

}

er.setModel(modelo);

} catch (SQLException ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

/\*\*

\* This method is called from within the constructor to initialize the form.

\* WARNING: Do NOT modify this code. The content of this method is always

\* regenerated by the Form Editor.

\*/

@SuppressWarnings("unchecked")

// <editor-fold defaultstate="collapsed" desc="Generated Code">

private void initComponents() {

jPanel1 = new javax.swing.JPanel();

jButton2 = new javax.swing.JButton();

jButton1 = new javax.swing.JButton();

jLabel1 = new javax.swing.JLabel();

jButton4 = new javax.swing.JButton();

jButton5 = new javax.swing.JButton();

buscar = new javax.swing.JTextField();

jScrollPane1 = new javax.swing.JScrollPane();

tbproduc = new javax.swing.JTable();

jButton3 = new javax.swing.JButton();

jButton7 = new javax.swing.JButton();

jButton6 = new javax.swing.JButton();

jScrollPane2 = new javax.swing.JScrollPane();

er = new javax.swing.JTable();

jLabel2 = new javax.swing.JLabel();

jLabel3 = new javax.swing.JLabel();

jLabel4 = new javax.swing.JLabel();

jLabel5 = new javax.swing.JLabel();

jLabel6 = new javax.swing.JLabel();

jLabel7 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT\_ON\_CLOSE);

setBackground(new java.awt.Color(0, 0, 255));

jPanel1.setBackground(new java.awt.Color(102, 102, 102));

jPanel1.setBorder(new javax.swing.border.MatteBorder(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\fon.jpg"))); // NOI18N

jPanel1.setToolTipText("");

jPanel1.setMaximumSize(getMaximumSize());

jPanel1.setLayout(null);

jButton2.setBackground(new java.awt.Color(204, 204, 204));

jButton2.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N

jButton2.setForeground(new java.awt.Color(0, 51, 255));

jButton2.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\vodka.jpg")); // NOI18N

jButton2.setText("VODKA ");

jButton2.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton2ActionPerformed(evt);

}

});

jPanel1.add(jButton2);

jButton2.setBounds(650, 120, 280, 170);

jButton1.setBackground(new java.awt.Color(204, 204, 204));

jButton1.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N

jButton1.setForeground(new java.awt.Color(0, 0, 255));

jButton1.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\coctel-cuba-libre-7612004.jpg")); // NOI18N

jButton1.setText("CUBA LIBRE ");

jButton1.setOpaque(false);

jButton1.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton1ActionPerformed(evt);

}

});

jPanel1.add(jButton1);

jButton1.setBounds(1060, 310, 285, 170);

jLabel1.setFont(new java.awt.Font("Algerian", 3, 80)); // NOI18N

jLabel1.setForeground(new java.awt.Color(0, 0, 204));

jLabel1.setText("ENIBRIATOR");

jPanel1.add(jLabel1);

jLabel1.setBounds(520, 10, 520, 100);

jLabel1.getAccessibleContext().setAccessibleDescription("");

jButton4.setBackground(new java.awt.Color(204, 204, 204));

jButton4.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N

jButton4.setForeground(new java.awt.Color(0, 0, 204));

jButton4.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\whisk.jpg")); // NOI18N

jButton4.setText("WHISKEY");

jButton4.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton4ActionPerformed(evt);

}

});

jPanel1.add(jButton4);

jButton4.setBounds(770, 310, 280, 170);

jButton5.setBackground(new java.awt.Color(204, 204, 204));

jButton5.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N

jButton5.setForeground(new java.awt.Color(0, 0, 204));

jButton5.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\paoma.jpg")); // NOI18N

jButton5.setText("PALOMA");

jButton5.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton5ActionPerformed(evt);

}

});

jPanel1.add(jButton5);

jButton5.setBounds(470, 310, 290, 170);

jPanel1.add(buscar);

buscar.setBounds(10, 60, 150, 30);

tbproduc.setBackground(new java.awt.Color(255, 153, 0));

tbproduc.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

},

new String [] {

}

));

jScrollPane1.setViewportView(tbproduc);

jPanel1.add(jScrollPane1);

jScrollPane1.setBounds(10, 160, 440, 140);

jButton3.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\BUS.jpg")); // NOI18N

jButton3.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton3ActionPerformed(evt);

}

});

jPanel1.add(jButton3);

jButton3.setBounds(170, 40, 150, 70);

jButton7.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\wheel\_all-shows.png")); // NOI18N

jButton7.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton7ActionPerformed(evt);

}

});

jPanel1.add(jButton7);

jButton7.setBounds(340, 20, 110, 120);

jButton6.setBackground(new java.awt.Color(204, 204, 204));

jButton6.setFont(new java.awt.Font("Tahoma", 1, 11)); // NOI18N

jButton6.setForeground(new java.awt.Color(0, 0, 255));

jButton6.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\caballiu.jpg")); // NOI18N

jButton6.setText("CABALLITO");

jButton6.addActionListener(new java.awt.event.ActionListener() {

public void actionPerformed(java.awt.event.ActionEvent evt) {

jButton6ActionPerformed(evt);

}

});

jPanel1.add(jButton6);

jButton6.setBounds(950, 120, 280, 170);

er.setBackground(new java.awt.Color(255, 153, 0));

er.setModel(new javax.swing.table.DefaultTableModel(

new Object [][] {

{},

{},

{},

{}

},

new String [] {

}

));

jScrollPane2.setViewportView(er);

jPanel1.add(jScrollPane2);

jScrollPane2.setBounds(10, 350, 430, 160);

jLabel2.setFont(new java.awt.Font("Ravie", 1, 12)); // NOI18N

jLabel2.setForeground(new java.awt.Color(255, 255, 255));

jLabel2.setText("DANIEL ISLAS CUERO");

jPanel1.add(jLabel2);

jLabel2.setBounds(820, 650, 230, 40);

jLabel3.setBackground(new java.awt.Color(153, 255, 255));

jLabel3.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N

jLabel3.setForeground(new java.awt.Color(0, 0, 255));

jLabel3.setText("UNIVERSIDAD PRIVADA DEL ESTADO DE MEXICO");

jPanel1.add(jLabel3);

jLabel3.setBounds(720, 680, 370, 20);

jLabel4.setFont(new java.awt.Font("Comic Sans MS", 1, 24)); // NOI18N

jLabel4.setForeground(new java.awt.Color(255, 153, 0));

jLabel4.setText("BEBIDAS HECHAS");

jPanel1.add(jLabel4);

jLabel4.setBounds(120, 310, 240, 34);

jLabel5.setIcon(new javax.swing.ImageIcon("C:\\Users\\DAN\\Desktop\\american-bar-nightflys-19to1.jpeg")); // NOI18N

jPanel1.add(jLabel5);

jLabel5.setBounds(-30, -10, 1480, 740);

jLabel6.setText("jLabel6");

jPanel1.add(jLabel6);

jLabel6.setBounds(530, 540, 160, 90);

jLabel7.setText("jLabel7");

jPanel1.add(jLabel7);

jLabel7.setBounds(600, 530, 34, 14);

javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());

getContentPane().setLayout(layout);

layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addGroup(layout.createSequentialGroup()

.addComponent(jPanel1, javax.swing.GroupLayout.PREFERRED\_SIZE, 1376, javax.swing.GroupLayout.PREFERRED\_SIZE)

.addGap(0, 42, Short.MAX\_VALUE))

);

layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

.addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT\_SIZE, 765, Short.MAX\_VALUE)

);

pack();

}// </editor-fold>

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

try {

Calendar calendario = Calendar.getInstance();

String e = "CUBA LIBRE";

int hora, min, seg, d, m, a;

d = calendario.get(Calendar.DATE);

m = calendario.get(Calendar.MONTH);

a = calendario.get(Calendar.YEAR);

hora = calendario.get(Calendar.HOUR\_OF\_DAY);

min = calendario.get(Calendar.MINUTE);

seg = calendario.get(Calendar.SECOND);

PreparedStatement pst = cn.prepareStatement("INSERT INTO he (nom, FECHA, HORA) VALUES (?,?,?)");

String c = "";

c = String.valueOf(d);

c= Integer.toString(d);

String x = "";

x = String.valueOf(m);

x= Integer.toString(m);

String o = "";

o = String.valueOf(a);

o= Integer.toString(a);

String i = "";

i = String.valueOf(hora);

i= Integer.toString(hora);

String q = "";

q = String.valueOf(min);

q= Integer.toString(min);

String z = "";

z = String.valueOf(seg);

z= Integer.toString(seg);

pst.setString(1, e);

pst.setString(2, c + "/" + x + "/" + o);

pst.setString(3, i + ":" + q + ":" + z);

pst.executeUpdate();

mostrardatos2("");

arduino.sendData("6");

} catch (Exception ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

try {

Calendar calendario = Calendar.getInstance();

String e = "VODKA";

int hora, min, seg, d, m, a;

d = calendario.get(Calendar.DATE);

m = calendario.get(Calendar.MONTH);

a = calendario.get(Calendar.YEAR);

hora = calendario.get(Calendar.HOUR\_OF\_DAY);

min = calendario.get(Calendar.MINUTE);

seg = calendario.get(Calendar.SECOND);

PreparedStatement pst = cn.prepareStatement("INSERT INTO he (nom, FECHA, HORA) VALUES (?,?,?)");

String c = "";

c = String.valueOf(d);

c= Integer.toString(d);

String x = "";

x = String.valueOf(m);

x= Integer.toString(m);

String o = "";

o = String.valueOf(a);

o= Integer.toString(a);

String i = "";

i = String.valueOf(hora);

i= Integer.toString(hora);

String q = "";

q = String.valueOf(min);

q= Integer.toString(min);

String z = "";

z = String.valueOf(seg);

z = Integer.toString(seg);

pst.setString(1, e);

pst.setString(2, c + "/" + x + "/" + o);

pst.setString(3, i + ":" + q + ":" + z);

pst.executeUpdate();

mostrardatos2("");

arduino.sendData("4");

} catch (Exception ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

try {

// TODO add your handling code here:

Calendar calendario = Calendar.getInstance();

String e = "WHISKEY";

int hora, min, seg, d, m, a;

d = calendario.get(Calendar.DATE);

m = calendario.get(Calendar.MONTH);

a = calendario.get(Calendar.YEAR);

hora = calendario.get(Calendar.HOUR\_OF\_DAY);

min = calendario.get(Calendar.MINUTE);

seg = calendario.get(Calendar.SECOND);

PreparedStatement pst = cn.prepareStatement("INSERT INTO he (nom, FECHA, HORA) VALUES (?,?,?)");

String c = "";

c = String.valueOf(d);

c= Integer.toString(d);

String x = "";

x = String.valueOf(m);

x= Integer.toString(m);

String o = "";

o = String.valueOf(a);

o= Integer.toString(a);

String i = "";

i = String.valueOf(hora);

i= Integer.toString(hora);

String q = "";

q = String.valueOf(min);

q= Integer.toString(min);

String z = "";

z = String.valueOf(seg);

z= Integer.toString(seg);

pst.setString(1, e);

pst.setString(2, c + "/" + x + "/" + o);

pst.setString(3, i + ":" + q + ":" + z);

pst.executeUpdate();

mostrardatos2("");

arduino.sendData("3");

} catch (Exception ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

mostrardatos(buscar.getText());

}

private void jButton7ActionPerformed(java.awt.event.ActionEvent evt) {

mostrardatos("");

}

private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {

try {

// TODO add your handling code here:

Calendar calendario = Calendar.getInstance();

String e = "PALOMA";

int hora, min, seg, d, m, a;

d = calendario.get(Calendar.DATE);

m = calendario.get(Calendar.MONTH);

a = calendario.get(Calendar.YEAR);

hora = calendario.get(Calendar.HOUR\_OF\_DAY);

min = calendario.get(Calendar.MINUTE);

seg = calendario.get(Calendar.SECOND);

PreparedStatement pst = cn.prepareStatement("INSERT INTO he (nom, FECHA, HORA) VALUES (?,?,?)");

String c = "";

c = String.valueOf(d);

c= Integer.toString(d);

String x = "";

x = String.valueOf(m);

x= Integer.toString(m);

String o = "";

o = String.valueOf(a);

o= Integer.toString(a);

String i = "";

i = String.valueOf(hora);

i= Integer.toString(hora);

String q = "";

q = String.valueOf(min);

q= Integer.toString(min);

String z = "";

z = String.valueOf(seg);

z= Integer.toString(seg);

pst.setString(1, e);

pst.setString(2, c + "/" + x + "/" + o);

pst.setString(3, i + ":" + q + ":" + z);

pst.executeUpdate();

mostrardatos2("");

arduino.sendData("2");

} catch (Exception ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

private void jButton6ActionPerformed(java.awt.event.ActionEvent evt) {

try {

// TODO add your handling code here:

Calendar calendario = Calendar.getInstance();

String e = "CABALLITO";

int hora, min, seg, d, m, a;

d = calendario.get(Calendar.DATE);

m = calendario.get(Calendar.MONTH);

a = calendario.get(Calendar.YEAR);

hora = calendario.get(Calendar.HOUR\_OF\_DAY);

min = calendario.get(Calendar.MINUTE);

seg = calendario.get(Calendar.SECOND);

PreparedStatement pst = cn.prepareStatement("INSERT INTO he (nom, FECHA, HORA) VALUES (?,?,?)");

String c = "";

c = String.valueOf(d);

c= Integer.toString(d);

String x = "";

x = String.valueOf(m);

x= Integer.toString(m);

String o = "";

o = String.valueOf(a);

o= Integer.toString(a);

String i = "";

i = String.valueOf(hora);

i= Integer.toString(hora);

String q = "";

q = String.valueOf(min);

q= Integer.toString(min);

String z = "";

z = String.valueOf(seg);

z= Integer.toString(seg);

pst.setString(1, e);

pst.setString(2, c + "/" + x + "/" + o);

pst.setString(3, i + ":" + q + ":" + z);

pst.executeUpdate();

mostrardatos2("");

arduino.sendData("1");

} catch (Exception ex) {

Logger.getLogger(Formulario.class.getName()).log(Level.SEVERE, null, ex);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String args[]) {

/\* Set the Nimbus look and feel \*/

//<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">

/\* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.

\* For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html

\*/

try {

for (javax.swing.UIManager.LookAndFeelInfo info : javax.swing.UIManager.getInstalledLookAndFeels()) {

if ("Nimbus".equals(info.getName())) {

javax.swing.UIManager.setLookAndFeel(info.getClassName());

break;

}

}

} catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Formulario.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/\* Create and display the form \*/

java.awt.EventQueue.invokeLater(new Runnable() {

public void run() {

new Formulario().setVisible(true);

}

});

}

// Variables declaration - do not modify

private javax.swing.JTextField buscar;

private javax.swing.JTable er;

private javax.swing.JButton jButton1;

private javax.swing.JButton jButton2;

private javax.swing.JButton jButton3;

private javax.swing.JButton jButton4;

private javax.swing.JButton jButton5;

private javax.swing.JButton jButton6;

private javax.swing.JButton jButton7;

private javax.swing.JLabel jLabel1;

private javax.swing.JLabel jLabel2;

private javax.swing.JLabel jLabel3;

private javax.swing.JLabel jLabel4;

private javax.swing.JLabel jLabel5;

private javax.swing.JLabel jLabel6;

private javax.swing.JLabel jLabel7;

private javax.swing.JPanel jPanel1;

private javax.swing.JScrollPane jScrollPane1;

private javax.swing.JScrollPane jScrollPane2;

private javax.swing.JTable tbproduc;

// End of variables declaration

conectar cc = new conectar();

Connection cn = cc.conexion();

}

## CONTROLADOR ARDUINO

#define VELOCIDAD 1000

int steps = 2;

int direccion = 3;

int reset = 4;

//para ron

long pasos = 114000;

long pasos3 = 33000;

long pasos4 = 81000;

//para vodka

long pasos2 = 38000;

long pasos5 = 19000;

long pasos6 = 19000;

//para whiski

long pasos7 = 57000;

long pasos8 = 38000;

//para equila

long pasos9 = 95000;

long pasos10 = 14000;

int input;

int motor1 = 10;

int motor2 = 9;

int motor3 = 8;

int motor4 = 7;

int motor5 = 6;

int motor6 = 5;

#include <MaxMatrix.h>

#include <avr/pgmspace.h>

PROGMEM unsigned char const CH[] = {

3, 8, B00000000, B00000000, B00000000, B00000000, B00000000, // space

1, 8, B01011111, B00000000, B00000000, B00000000, B00000000, // !

3, 8, B00000011, B00000000, B00000011, B00000000, B00000000, // "

5, 8, B00010100, B00111110, B00010100, B00111110, B00010100, // #

4, 8, B00100100, B01101010, B00101011, B00010010, B00000000, // $

5, 8, B01100011, B00010011, B00001000, B01100100, B01100011, // %

5, 8, B00110110, B01001001, B01010110, B00100000, B01010000, // &

1, 8, B00000011, B00000000, B00000000, B00000000, B00000000, // '

3, 8, B00011100, B00100010, B01000001, B00000000, B00000000, // (

3, 8, B01000001, B00100010, B00011100, B00000000, B00000000, // )

5, 8, B00101000, B00011000, B00001110, B00011000, B00101000, // \*

5, 8, B00001000, B00001000, B00111110, B00001000, B00001000, // +

2, 8, B10110000, B01110000, B00000000, B00000000, B00000000, // ,

4, 8, B00001000, B00001000, B00001000, B00001000, B00000000, // -

2, 8, B01100000, B01100000, B00000000, B00000000, B00000000, // .

4, 8, B01100000, B00011000, B00000110, B00000001, B00000000, // /

4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // 0

3, 8, B01000010, B01111111, B01000000, B00000000, B00000000, // 1

4, 8, B01100010, B01010001, B01001001, B01000110, B00000000, // 2

4, 8, B00100010, B01000001, B01001001, B00110110, B00000000, // 3

4, 8, B00011000, B00010100, B00010010, B01111111, B00000000, // 4

4, 8, B00100111, B01000101, B01000101, B00111001, B00000000, // 5

4, 8, B00111110, B01001001, B01001001, B00110000, B00000000, // 6

4, 8, B01100001, B00010001, B00001001, B00000111, B00000000, // 7

4, 8, B00110110, B01001001, B01001001, B00110110, B00000000, // 8

4, 8, B00000110, B01001001, B01001001, B00111110, B00000000, // 9

2, 8, B01010000, B00000000, B00000000, B00000000, B00000000, // :

2, 8, B10000000, B01010000, B00000000, B00000000, B00000000, // ;

3, 8, B00010000, B00101000, B01000100, B00000000, B00000000, // <

3, 8, B00010100, B00010100, B00010100, B00000000, B00000000, // =

3, 8, B01000100, B00101000, B00010000, B00000000, B00000000, // >

4, 8, B00000010, B01011001, B00001001, B00000110, B00000000, // ?

5, 8, B00111110, B01001001, B01010101, B01011101, B00001110, // @

4, 8, B01111110, B00010001, B00010001, B01111110, B00000000, // A

4, 8, B01111111, B01001001, B01001001, B00110110, B00000000, // B

4, 8, B00111110, B01000001, B01000001, B00100010, B00000000, // C

4, 8, B01111111, B01000001, B01000001, B00111110, B00000000, // D

4, 8, B01111111, B01001001, B01001001, B01000001, B00000000, // E

4, 8, B01111111, B00001001, B00001001, B00000001, B00000000, // F

4, 8, B00111110, B01000001, B01001001, B01111010, B00000000, // G

4, 8, B01111111, B00001000, B00001000, B01111111, B00000000, // H

3, 8, B01000001, B01111111, B01000001, B00000000, B00000000, // I

4, 8, B00110000, B01000000, B01000001, B00111111, B00000000, // J

4, 8, B01111111, B00001000, B00010100, B01100011, B00000000, // K

4, 8, B01111111, B01000000, B01000000, B01000000, B00000000, // L

5, 8, B01111111, B00000010, B00001100, B00000010, B01111111, // M

5, 8, B01111111, B00000100, B00001000, B00010000, B01111111, // N

4, 8, B00111110, B01000001, B01000001, B00111110, B00000000, // O

4, 8, B01111111, B00001001, B00001001, B00000110, B00000000, // P

4, 8, B00111110, B01000001, B01000001, B10111110, B00000000, // Q

4, 8, B01111111, B00001001, B00001001, B01110110, B00000000, // R

4, 8, B01000110, B01001001, B01001001, B00110010, B00000000, // S

5, 8, B00000001, B00000001, B01111111, B00000001, B00000001, // T

4, 8, B00111111, B01000000, B01000000, B00111111, B00000000, // U

5, 8, B00001111, B00110000, B01000000, B00110000, B00001111, // V

5, 8, B00111111, B01000000, B00111000, B01000000, B00111111, // W

5, 8, B01100011, B00010100, B00001000, B00010100, B01100011, // X

5, 8, B00000111, B00001000, B01110000, B00001000, B00000111, // Y

4, 8, B01100001, B01010001, B01001001, B01000111, B00000000, // Z

2, 8, B01111111, B01000001, B00000000, B00000000, B00000000, // [

4, 8, B00000001, B00000110, B00011000, B01100000, B00000000, // \ backslash

2, 8, B01000001, B01111111, B00000000, B00000000, B00000000, // ]

3, 8, B00000010, B00000001, B00000010, B00000000, B00000000, // hat

4, 8, B01000000, B01000000, B01000000, B01000000, B00000000, // \_

2, 8, B00000001, B00000010, B00000000, B00000000, B00000000, // `

4, 8, B00100000, B01010100, B01010100, B01111000, B00000000, // a

4, 8, B01111111, B01000100, B01000100, B00111000, B00000000, // b

4, 8, B00111000, B01000100, B01000100, B00101000, B00000000, // c

4, 8, B00111000, B01000100, B01000100, B01111111, B00000000, // d

4, 8, B00111000, B01010100, B01010100, B00011000, B00000000, // e

3, 8, B00000100, B01111110, B00000101, B00000000, B00000000, // f

4, 8, B10011000, B10100100, B10100100, B01111000, B00000000, // g

4, 8, B01111111, B00000100, B00000100, B01111000, B00000000, // h

3, 8, B01000100, B01111101, B01000000, B00000000, B00000000, // i

4, 8, B01000000, B10000000, B10000100, B01111101, B00000000, // j

4, 8, B01111111, B00010000, B00101000, B01000100, B00000000, // k

3, 8, B01000001, B01111111, B01000000, B00000000, B00000000, // l

5, 8, B01111100, B00000100, B01111100, B00000100, B01111000, // m

4, 8, B01111100, B00000100, B00000100, B01111000, B00000000, // n

4, 8, B00111000, B01000100, B01000100, B00111000, B00000000, // o

4, 8, B11111100, B00100100, B00100100, B00011000, B00000000, // p

4, 8, B00011000, B00100100, B00100100, B11111100, B00000000, // q

4, 8, B01111100, B00001000, B00000100, B00000100, B00000000, // r

4, 8, B01001000, B01010100, B01010100, B00100100, B00000000, // s

3, 8, B00000100, B00111111, B01000100, B00000000, B00000000, // t

4, 8, B00111100, B01000000, B01000000, B01111100, B00000000, // u

5, 8, B00011100, B00100000, B01000000, B00100000, B00011100, // v

5, 8, B00111100, B01000000, B00111100, B01000000, B00111100, // w

5, 8, B01000100, B00101000, B00010000, B00101000, B01000100, // x

4, 8, B10011100, B10100000, B10100000, B01111100, B00000000, // y

3, 8, B01100100, B01010100, B01001100, B00000000, B00000000, // z

3, 8, B00001000, B00110110, B01000001, B00000000, B00000000, // {

1, 8, B01111111, B00000000, B00000000, B00000000, B00000000, // |

3, 8, B01000001, B00110110, B00001000, B00000000, B00000000, // }

4, 8, B00001000, B00000100, B00001000, B00000100, B00000000, // ~

};

int data = 13; // DIN pin del modulo MAX7219

int load = 12; // CS pin del modulo MAX7219

int clock = 11; // CLK pin del modulo MAX7219

int maxInUse = 5; //Cambie este valor dependiendo del tipo de modulo MAX7219 que use

MaxMatrix m(data, load, clock, maxInUse); // Define el modulo

byte buffer[10];

char mensaje1[] = " ...PONGA EL VASO... "; //Escriba el mensaje a desplegar

char mensaje2[] = "ESPERE..";

char mensaje3[] = "..TOME EL VASO..";

void setup(){

pinMode(steps, OUTPUT);

pinMode(direccion, OUTPUT);

pinMode(reset, OUTPUT);

pinMode(motor1, OUTPUT);

pinMode(motor2, OUTPUT); // Declaramos que utilizaremos el pin 13 como salida

pinMode(motor3, OUTPUT);

pinMode(motor4, OUTPUT);

pinMode(motor5, OUTPUT);

pinMode(motor6, OUTPUT);

pinMode(13,OUTPUT); //Conexion a DIN

pinMode(12,OUTPUT); //Conexion a CS

pinMode(11,OUTPUT); //Conexion a CLK

m.init(); // inicializa el modulo

m.setIntensity(5);

Serial.begin(9600);

}

void loop(){

printStringWithShift(mensaje1, 40); // El ultimo termino se usa para la velocidad del mensaje

if (Serial.available()>0){

input=Serial.read();

int onTime = 1700; // licor de 1 onza

int onTime2 = 4500; // jugo o refresco

int onTime3 = 1800; // RON

int offTime = 1000;

// ron refresco

if (input=='6'){

printStringWithShift(mensaje2, 40);

//paso

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, HIGH);

for (int i = 0; i<pasos; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, HIGH); // This LOW to HIGH change is what creates the

digitalWrite(steps, LOW); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

//acaba pasos

digitalWrite(motor6, HIGH);

delay (onTime3);

digitalWrite(motor6, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos3; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

digitalWrite(motor4, HIGH);

delay (onTime2);

digitalWrite(motor4, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos4; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

printStringWithShift(mensaje3, 40);

}

//vodka

if (input=='4'){

printStringWithShift(mensaje2, 40);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, HIGH);

for (int i = 0; i<pasos2; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, HIGH); // This LOW to HIGH change is what creates the

digitalWrite(steps, LOW); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

digitalWrite(motor2, HIGH);

delay (onTime);

digitalWrite(motor2, LOW);

delay (offTime);//Si el valor de input es diferente de 1, se apaga el LED

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos5; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

digitalWrite(motor1, HIGH);

delay (onTime2);

digitalWrite(motor1, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos6; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

printStringWithShift(mensaje3, 40);

}

// whiski

if (input=='3'){

printStringWithShift(mensaje2, 40);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, HIGH);

for (int i = 0; i<pasos7; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, HIGH); // This LOW to HIGH change is what creates the

digitalWrite(steps, LOW); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

digitalWrite(motor3, HIGH);

delay (onTime);

digitalWrite(motor3, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos8; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

digitalWrite(motor1, HIGH);

delay (onTime2);

digitalWrite(motor1, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos6; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

printStringWithShift(mensaje3, 40);

}

if (input=='2'){

printStringWithShift(mensaje2, 40);

//paso

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, HIGH);

for (int i = 0; i<pasos9; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, HIGH); // This LOW to HIGH change is what creates the

digitalWrite(steps, LOW); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

//acaba pasos

digitalWrite(motor5, HIGH);

delay (onTime3);

digitalWrite(motor5, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos10; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

digitalWrite(motor4, HIGH);

delay (onTime2);

digitalWrite(motor4, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos4; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

printStringWithShift(mensaje3, 40);

}

if (input=='1'){

printStringWithShift(mensaje2, 40);

//paso

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, HIGH);

for (int i = 0; i<pasos9; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, HIGH); // This LOW to HIGH change is what creates the

digitalWrite(steps, LOW); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

digitalWrite(motor5, HIGH);

delay (onTime3);

digitalWrite(motor5, LOW);

delay (offTime);

digitalWrite(reset, LOW); //Mientras reset este en LOW el motor permanecerá apagado y no sufrirá. El chip apagará todos los puertos y no leerá comandos.

delay(1000);

digitalWrite(reset, HIGH); //Cuando reset se encuentre en HIGH el motor arrancará y leerá los comandos enviados.

digitalWrite(direccion, LOW);

for (int i = 0; i<pasos9; i++) //Equivale al numero de vueltas (200 es 360º grados) o micropasos

{

digitalWrite(steps, LOW); // LOW to HIGH hace que el motor avance ya que da la orden

digitalWrite(steps, HIGH); // al A4988 de avanzar una vez por cada pulso de energia.

delayMicroseconds(VELOCIDAD); // Regula la velocidad, cuanto mas bajo mas velocidad.

}

printStringWithShift(mensaje3, 40);

}

}

}

//mensaje

void printCharWithShift(char c, int shift\_speed){ // Imprime caracteres

if (c < 32) return;

c -= 32;

memcpy\_P(buffer, CH + 7\*c, 7);

m.writeSprite(maxInUse\*8, 0, buffer);

m.setColumn(maxInUse\*8 + buffer[0], 0);

for (int i=0; i<buffer[0]+1; i++)

{

delay(shift\_speed);

m.shiftLeft(false, false);

}

}

void printStringWithShift(char\* s, int shift\_speed){ // Imprime cadena de caracteres

while (\*s != 0){

printCharWithShift(\*s, shift\_speed);

s++;

}

}

void printString(char\* s) // Imprime cadena

{

int col = 0;

while (\*s != 0)

{

if (\*s < 32) continue;

char c = \*s - 32;

memcpy\_P(buffer, CH + 7\*c, 7);

m.writeSprite(col, 0, buffer);

m.setColumn(col + buffer[0], 0);

col += buffer[0] + 1;

s++;

}

}

