CREACIÓN E IMPLEMENTACIÓN DE TALLERES ESCOLARES DE INICIACIÓN A LA ELECTRÓNICA, PROGRAMACIÓN EN SISTEMAS Y MICROCONTROLADORES CON ARDUINO

PAOLA ARIZA COLPAS

libro digital

CUC
UNIVERSIDAD DE LA COSTA
1970
VIGILADA MINEDUCACIÓN

Creación e implementacion de talleres escolares de iniciación a la electrónica, programación en sistemas y microcontroladores con Arduino

Ariza Colpas, Paola
Creación e implementación de talleres escolares de iniciación a la electrónica, programas en sistemas y microcontroladores con Arduino / Paola Ariza Colpas. – Barranquilla: Corporación Universidad de la Costa, 2021

ISBN: 978-958-5172-18-0 (digital)
ISBN: 978-958-8710-81-5 (impreso)

Educación
Talleres escolares – Organización
Enseñanza y tecnología

373.2 A719

Creación e implementacion de talleres escolares de iniciación a la electrónica, programación en sistemas y microcontroladores con Arduino

Paola Ariza Colpas

UNIVERSIDAD
DE LA COSTA
1970

2011

Creación e implementacion de talleres escolares de iniciación a la electrónica, programación en sistemas y micro-controladores con Arduino

Autores:
Paola Ariza Colpas

Primera edición digital, 2021©
Primera edición impresa, 2010©

ISBN: 978-958-5172-18-0 (digital)
ISBN: 978-958-8710-81-5 (impreso)

Corporación Universidad de la Costa, CUC
Barranquilla - Colombia

Editorial Universitaria de la Costa S.A.S.
Calle 58 No. 55-66
Teléfono: (575) 344 4623
educosta@cuc.edu.co

Lauren J. Castro Bolaño
Directora (2021)

Clara Inés De la Roche
Coordinación Editorial (2011)

Hecho el depósito que exige la ley

FUNDADORES
CORPORACIÓN UNIVERSITARIA DE LA COSTA CUC

EDUARDO CRISSIEN SAMPER

RUBÉN MAURY PERTUZ (q.e.p.d.)

NULVIA BORRERO HERRERA

MARÍA ARDILA DE MAURY

RAMIRO MORENO NORIEGA

RODRIGO NIEBLES DE LA CRUZ (q.e.p.d.)

MIGUEL ANTEQUERA STAND

PERSONAL DIRECTIVO
CORPORACIÓN UNIVERSITARIA DE LA COSTA CUC

ITO JOSÉ CRISSIEN BORRERO
Rector

MARIO MAURY ARDILA
Director Departamento de Posgrados

CAROLINA PADILLA VILLA
Secretaria General

GLORIA CECILIA MORENO GÓMEZ
Vicerrectora Académica

HENRY MAURY ARDILA
Vicerrector de Investigaciones

JOSÉ EDUARDO CRISSIEN ORELLANO (e)
Vicerrector de Extensión

JAIME DÍAZ ARENAS
Vicerrector Administrativo

RODOLFO MAURY ARDILA
Vicerrector de Bienestar

HERNANDO ANTEQUERA MANOTAS
Vicerrector Financiero

ALFREDO GÓMEZ VILLANUEVA
Decano Facultad de Arquitectura

JAVIER MORENO JUVINAO
Decano Facultad de Ciencias Económicas

ALFREDO PEÑA SALOM
Decano Facultad de Derecho

MILDRED PUELLO SCARPATI
Decana Facultad de Psicología

FAIRUZ VIOLET OSPINO VALDIRIS
Decana Facultad de Ingeniería

NADIA JUDITH OLAYA CORONADO
Decana Facultad de Ciencias Ambientales

Agradecimientos

Este libro pudo desarrollarse gracias a la ayuda incondicional de todo un equipo de trabajo que involucra a personal directivo, administrativo y académico de la Corporación Universitaria de la Costa – CUC. Entre ellos se destaca al consejo de fundadores de la institución, conformado por Eduardo Crissien Samper, Rubén Maury Pertuz (q.e.p.d.), Nulvia Borrero Barraza, María Ardila De Maury, Ramiro Moreno Noriega, Rodrigo Niebles De la Cruz (q.e.p.d.) y Miguel Antequera Stand. A Tito José Crissien Borrero, Rector general; a Gloria Cecilia Moreno, Vicerrectora académica; a Henry Maury Ardila, Vicerrector de investigación; a Fairuz Ospino Valdiris, Decana de la Facultad de Ingenierías; a Perla Blanco Miranda, Coordinadora de Educosta; y a todos aquellos que hicieron sus aportaciones técnicas y de estilo, las cuales invitaron a la reflexión, afinamiento y fortalecimiento de ésta obra.

Paola Ariza

Tabla de Contenido

Tabla de Figuras

Tabla de Código

Índice de Tablas

Introducción al proyecto de extensión

Hoy por hoy, la articulación con la educación media es uno de los aspectos más importantes para diferentes instituciones de educación superior, sobre todo para captar el interés de los potenciales estudiantes a la vinculación de carreras profesionales de su afinidad.

Por esta razón, es meritorio que los estudiantes de los colegios distritales puedan tener contacto con la electrónica y los lenguajes de programación como ejes temáticos de vanguardia ocupacional. Este proyecto, por lo tanto, se encuentra enfocado en brindar la orientación necesaria a estos potenciales estudiantes para el proceso de elección de estos temas afines.

Se ha identificado la placa Arduino como base fundamental para la enseñanza de la electrónica y los lenguajes de programación a los estudiantes de la media vocacional. Es, por eso, la principal finalidad de este proyecto de investigación la consecución de talleres escolares de iniciación a la electrónica, programación en sistemas y microcontroladores con Arduino en los colegios Distritales del centro histórico de Barranquilla como apoyo a la adquisición de estos conocimientos por parte de los estudiantes y su acercamiento al proceso de elección de una carrera profesional a cursar.

Para esta investigación fueron definidos los siguientes objetivos:

General:

Crear e implementar talleres escolares de iniciación a la electrónica, programación en sistemas y microcontroladores con Arduino en los colegios Distritales del centro histórico de Barranquilla.

Específicos:

- Analizar el estado actual de la enseñanza de electrónica, programación en sistemas y microcontroladores en los colegios distritales del centro histórico de Barranquilla.
- Diseñar talleres amigables para la enseñanza de Arduino en los colegios distritales del centro histórico de Barranquilla.
- Diseñar cartillas de aprendizaje para la enseñanza de Arduino en los colegios distritales del centro histórico de Barranquilla
- Implementar talleres de aprendizaje de Arduino en los colegios distritales del centro histórico de Barranquilla.

Como respuesta a este proyecto de extensión social, la Corporación Universitaria de la Costa CUC, seleccion**ó** entre los colegios distritales del centro histórico de Barranquilla a los siguientes:

Posteriormente, se estableció la semana de octubre 11 a octubre 15 de 2011 para que en el horario de 8:00 a.m. a 12:00 m fueran realizadas estas capacitaciones a docentes distritales con la finalidad de que éstos se convirtieran en transmisores y retroalimentadores de este conocimiento a sus estudiantes.

Se decidió el siguiente cronograma de actividades para el desarrollo de taller de iniciación en Arduino.

Tabla No 1.
Temáticas de los talleres de Arduino 2011

No	Temática
1	Introducción
2	Hardware y software libre
3	Microcontroladores y procesadores

4	Conociendo Arduino
5	LED
6	Resistencias
7	Voltaje AC DC
8	Intensidad
9	Watts
10	Voltímetro (Medir voltaje, resistencia, etc.)
11	Switch / Pulsadores
12	Variables
13	Manejo de puerto serial USB con Arduino
14	Puertos análogos y digitales (PWM)
15	Manejo de memoria
16	EEPROM, FLASH, SRAM
17	Bit, byte, kilobyte
18	LDR (Foto resistencia)
19	Potenciómetro
20	Relay
21	Transistor
22	Diodo
23	Piezo
24	LDR
25	Encendiendo LEDs
26	LEDs Parpadeantes
27	Secuenciador con LEDs

Los talleres de iniciación en Arduino fueron desarrollados en las instalaciones de la Corporación Universitaria de la Costa - CUC. En el anexo A se muestran fotografías del desarrollo de esta actividad en la institución.

Información general

Arduino es una plataforma educativa para la enseñanza de tecnología que incluye conceptos de sistemas, programación, electrónica básica, automatización y robótica, entre otros.

Arduino es una plataforma de hardware y software de código abierto (libre) basada en una sencilla placa con entradas y salidas, analógicas y digitales. Es un dispositivo que conecta el mundo físico con el mundo virtual, o el mundo analógico con el digital.

Los talleres están orientados a estudiantes, profesionales, educadores y artistas con interés en adquirir los conocimientos en electrónica digital y análoga y programación de una forma rápida y divertida. Ya que se trata de un taller de iniciación, no es necesario tener conocimientos previos.

Apaga y prende LED N veces

Observación: esta práctica ha sido recogida de la página oficial de Arduino, http://www.arduino.cc.

Como su nombre lo indica, esta práctica se trata de prender y apagar un número determinado de veces un LED; un número que se indicará por medio del serial monitor que encontramos en nuestra IDE de Arduino.

Figura No 1. Arduino.

Para el programa se utilizarán las funciones del puerto serie[1]:

[1] Jorge Pomares Baeza. Manual de Arduino. Universidad de Alicante. Disponible en: http://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf

- Serial.begin(rate): abre un puerto serie y especifica la velocidad de transmisión. La velocidad típica para comunicación en el ordenador es de 9600, aunque se pueden soportar otras velocidades.
- Serial.println(data): imprime los datos al puerto serie seguido por un retorno de línea automático. Este comando tiene la misma forma que Serial.print( ), la diferencia es que Serial.Println tiene el salto de línea al final. Este comando puede ser utilizado para la depuración de programas.

Para ello se pueden enviar mensajes de depuración y valores de variables por el puerto serie.

Para tener en cuenta: el puerto serie y el serial monitor deben estar configurados a la misma velocidad.

- Serial.read( ): Lee o captura un byte, o sea, un carácter desde el puerto serie. Devuelve -1 si no hay ningún carácter en el puerto serie.
- Serial.available( ): Devuelve el número de caracteres disponibles para leer desde el puerto serie.

Esquema:

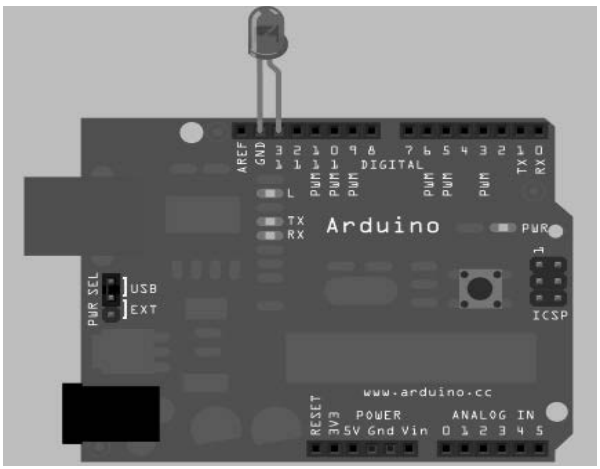

Figura No 2. Placa Arduino.

En nuestro IDE de Arduino:

```
int ledPin = 13; // selecciona el pin para el led

int val = 0; // variable que almacena el valor leído del puerto

void setup() {

pinMode(ledPin,OUTPUT); // declaración del modo del LED como salida

Serial.begin(19200); // conecta con el puerto serie a la velocidad de 19200

Serial.println("Bienvenido al monitor serial donde veras como se controla un LED");}

void loop () {

val = Serial.read(); // lee el número del puerto (una sola cifra)

//si el valor leído es un solo digito se ejecuta el programa

if (val > '0' && val <= '9' ) {

val = val - '0'; // convierte el carácter leído en un numero

for(int i=0; i<val; i++) {

Serial.println("Led Encendido");

digitalWrite(ledPin,HIGH); // enciende el LED

delay(150); // Espera 150 milisegundos

Serial.println("Led Apagado");

digitalWrite(ledPin, LOW); // apaga el LED

delay(150); // espera } }}
```

Código Fuente No. 1

Al subir nuestro programa a la Board obtendremos lo siguiente: insertamos un número y le damos send (enviar).

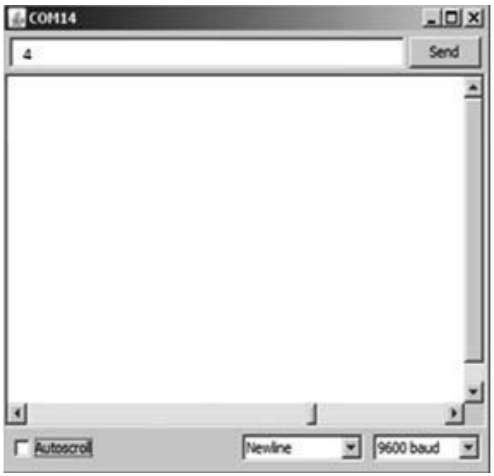

Figura No 3. Software de Arduino.

Lo cual trae como resultado:

Figura No 4. Ejecución de encendido y apagado de los LED.

Aprendiendo variables en Arduino

Variable

Una variable es una posición en memoria en la cual se pueden guardar datos. Debe estar inicializada y declarada como el tipo de dato que se almacenará y, opcionalmente, asignada a un determinado valor. Puede nombrarse como nos guste siempre y cuando no sea una palabra reservada del IDE de Arduino.

Ejemplo: Declaremos una variable llamada Num y otra llamada letra.

```
Int Num;
// Se declaró la variable de tipo entero
string letra;
// Se declaró la variable de tipo carácter
```

Código Fuente No. 2

EjemploDeclaremos una variable llamada Num asignándole un determinado valor.

```
Int Num = 0;
// Se declaró la variable de tipo entero
// La variable tiene un valor inicial de 0
```

Código Fuente No. 3

Ejemplo: Declaremos una variable llamada letra asignándole un determinado valor.

```
string letra = "a";
// Se declaró la variable de tipo string
// La variable tiene un dato inicial "a"
```

Código Fuente No. 4

Ejemplo: Después de declarada la variable Num se le asignara otro valor.

Num = 4;

// A la variable de tipo entero se le pueden asignar valores, sin importar que no tenga valor inicial.

Ejemplo: Después de declarada la variable Num y asignarle un valor de 0, se le asignara otro valor.

Num = 4;

// A la variable de tipo entero se le pueden asignar valores, sin importar qué dato tenga como valor inicial.

Las variables deben tomar nombres descriptivos para hacer el código más legible. Nombres de variables pueden ser "contactoSensor" o "pulsador", para ayudar al programador y a cualquier otra persona a leer el código y entender lo que representa la variable.

Una vez que una variable ha sido asignada, o re-asignada, como vimos en los ejemplos anteriores, podemos probar su valor para ver si cumple ciertas condiciones (instrucciones tipo if..) o puede utilizar directamente su valor.

Vamos a realizar varios ejercicios en nuestro IDE de Arduino y compilamos para que no existan errores.

Prueba 1:

Si la variable "Num" es inferior a 100, se le asigna el valor 100 a "Num" y, a continuación, establece un retardo (delay) utilizando como valor "Num", que ahora será como mínimo de valor 100:

```
Num = 90
Si Num < 100, entonces,
haga
Num = 100
Retardo (Num)
Fin
```

En nuestro IDE sería:

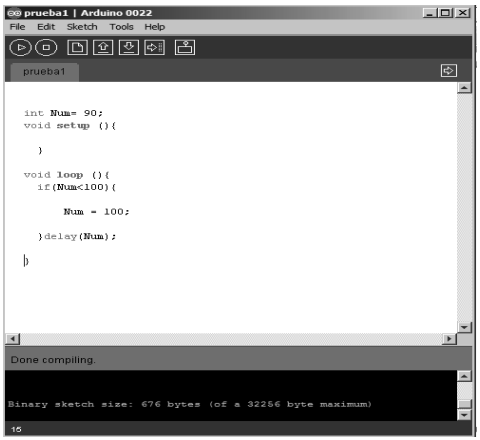

Figura No 5. Probando Arduino.

Done compiling, significa que no hay errores.

Prueba 2:

Mientras que la variable "Num" sea mayor a 10 o menor a 50, se le asigna un incremento a la variable "Num" y, a continuación, a Num se le asignara un valor inicial de 12:

```
Num = 12
Mientras que Num sea mayor a 10 o menor de 50, entonces:
Haga
Num = num + 1
Fin
```

En nuestro IDE sería:

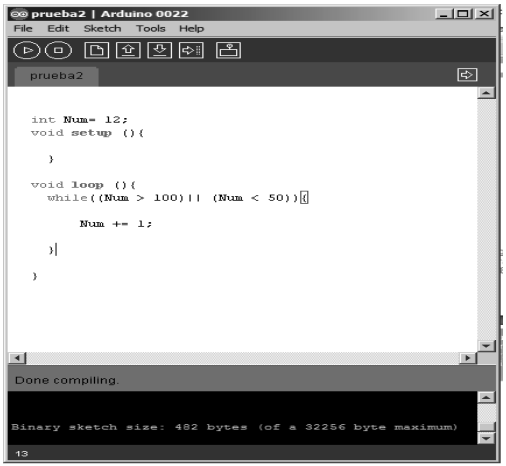

Figura No 6. Ciclos en Arduino.

Prueba 3:

Con la placa Arduino, trabajaremos la variable pin con una asignación del pin 13 e insertaremos un LED en el pin 12.

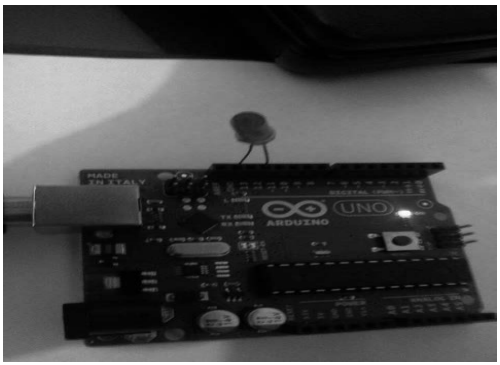

Figura No 7. Ejecución en la placa Arduino

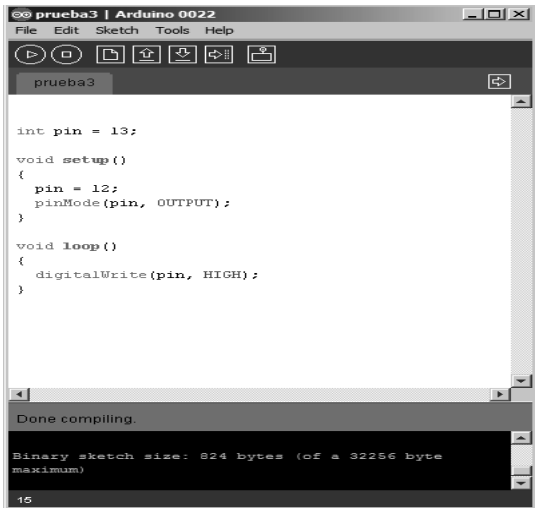

Figura No 8. Configurando PIN 12.

Este programa se sube a la placa y debe encenderse como se muestra a continuación:

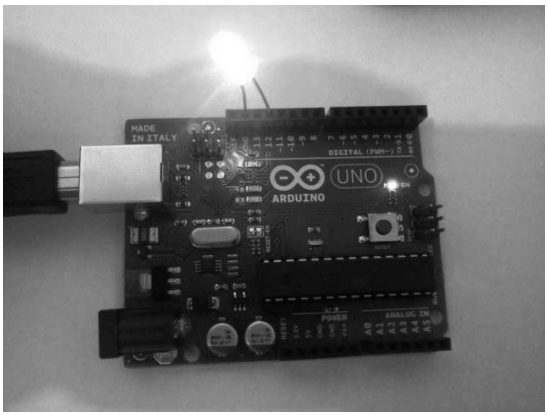

Figura No 9. Ejecución en la placa Arduino.

Aprendiendo intensidad de la corriente

¿Qué es la corriente?

La corriente eléctrica es aquella que representa el flujo de electrones de un punto de mayor potencial eléctrico a otro de menor potencial, en otras palabras, es la cantidad que nos permite medir el movimiento de una agrupación mayor de electrones hacia otra agrupación de menor cantidad.

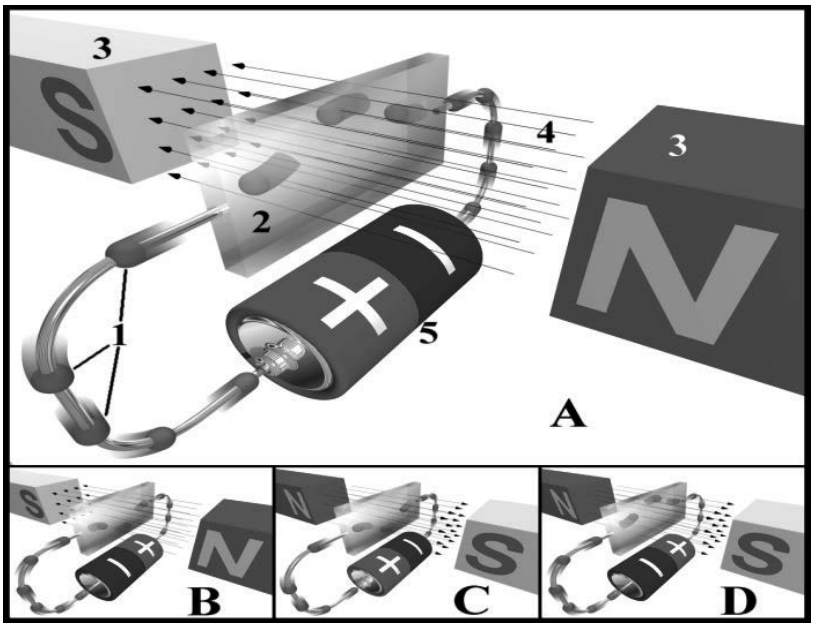

Figura No 10. Corriente eléctrica.[2]

Se puede hacer una analogía de un circuito eléctrico con la siguiente imagen:

[2] Tomado de: http://upload.wikimedia.org/wikipedia/commons/0/01/Hall_effect.png

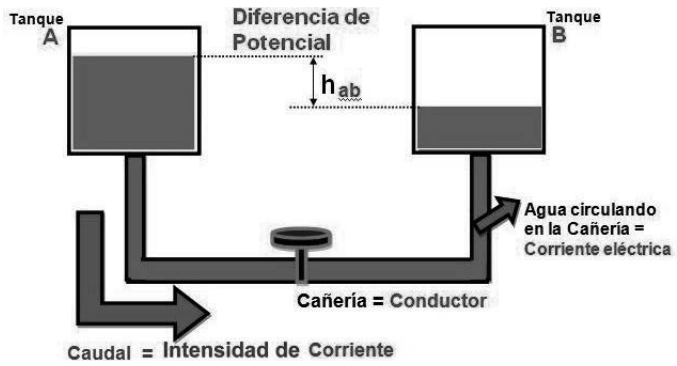

Figura No 11. Diferencia de potencial.[3]

Si imaginamos el caso en el que la llave esté abierta, el paso de agua del tanque A hacia el tanque B a través de la cañería, representa la corriente eléctrica. Entonces el caudal de agua equivale a la intensidad de corriente eléctrica.

Unidad de medición

Su unidad de medición es el Amperio, representado por la letra (A).

Figura No 12. Unidad de medición.[4]

[3] Tomado de: http://3.bp.blogspot.com/_sLlf8160jQw/Sw7KfSwlUVI/AAAAAAAAACI/XBAAoiLdHCs/s1600/corriente.jpg

[4] Tomado de: http://motosdecoleccion.com/tienda/images/Amperimetro12A50%20Bl.jpg

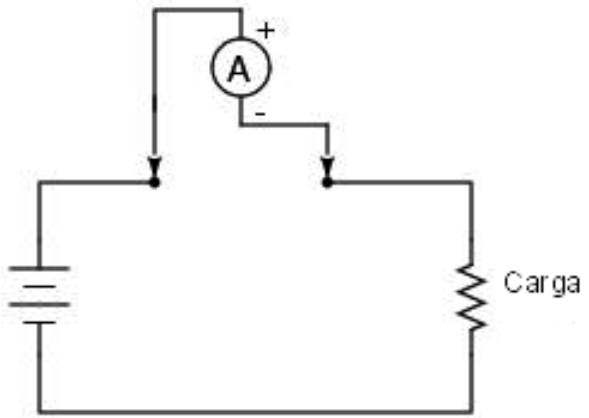

Figura No 13. Unidad de medición en el circuito.[5]

Nota: en la medición de corriente en los circuitos, ésta debe hacerse manteniendo contacto con uno de los terminales del elemento medidor (amperímetro) y el elemento a medir (generalmente una resistencia), y la otra terminal se conecta al circuito armado en puente de conexión entre el circuito y una de las terminales del elemento a medir, ésta se denomina en serie y la podemos observar en la figura anterior.

Ley de Ohm

La definición matemática que es usualmente usada en los circuitos electrónicos para realizar cálculos y diseñar circuitos según la corriente necesitada es la siguiente:

$$Intensidad\ el[?][?]ctrica = \frac{Voltaje\ de\ la\ resistencia}{Resistencia}$$

[5] Tomado de: http://imerl.fing.edu.uy/calcu3/Curso/Multimedia/Pinza%20Amperim%C3%A9trica_archivos/image001.gif

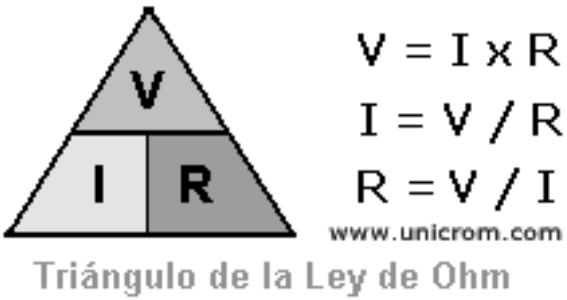

Figura No 14. Ley de Ohm.[6]

Corriente alterna

La corriente alterna es aquella que varía su intensidad y dirección del flujo de los electrones a través del tiempo. Ésta es la que comúnmente es generada por las centrales eléctricas y distribuida a cada una de nuestras casas para suministrar energía eléctrica a cada uno de los electrodomésticos. Es importante precisar que esos cambios de intensidad y flujo suelen ser cíclicos, generando frecuencias de oscilación en dicha magnitud, siendo para el estándar europeo 50 Hz y el estándar Americano 60 Hz, el cual es utilizado en Colombia.

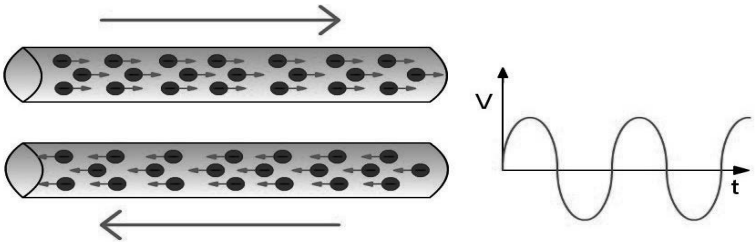

Figura No 15. Corriente Alterna.[7]

[6] Electrónica Unicrom. Disponible en: http://imagenes.unicrom.com/triangulo_ley_ohm.gif

[7] Corriente Alterna. Disponible en: http://2.bp.blogspot.com/-6w1AN1vKSh4/TcdEXTA510I/AAAAAAAAACc/9iIrZxJTr8A/s1600/corriente_alterna.jpg

Corriente continua o directa

La corriente continua es aquella que mantiene su intensidad y dirección del flujo de los electrones a través del tiempo. Las baterías producen corriente directa dentro de un circuito debido a que sus terminales siempre poseen el mismo signo de carga por el cual circulan los electrones.

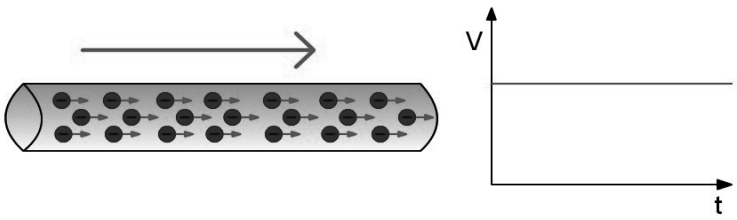

Figura No 16. Corriente Continua.[8]

Aprendiendo voltaje AC-DC

¿Qué es el voltaje?

La definición matemática que es usualmente usada en los circuitos electrónicos para realizar cálculos y diseñar circuitos según el voltaje necesitado es la siguiente:

Voltaje AC

Se puede definir como voltaje alterno aquel que varía de manera continua su amplitud (voltaje) y su polaridad a través del tiempo. Llamamos polaridad al cambio de dirección del flujo de los electrones. La hay de dos tipos polaridad positiva y negativa, y existe un tercer estado nulo sin polaridad.

[8] Corriente Continua. Disponible en: http://www.portaleso.com/usuarios/Toni/web_magnetismo_3/imagenes/corriente_continua.jpg

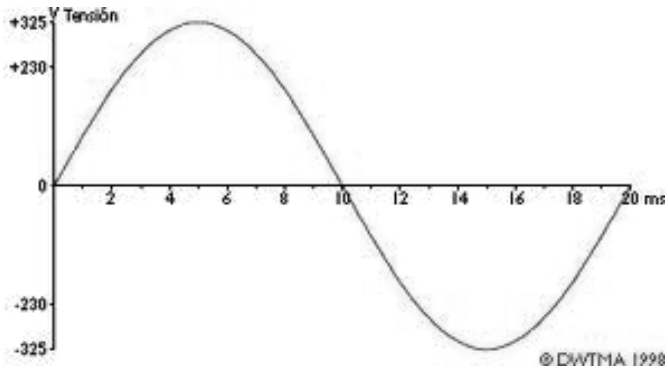

Figura No 17. Tensión Alterna.[9]

Voltaje DC

Es aquella tensión que mantiene sus características de amplitud y polaridad de forma constante a través del tiempo. Este voltaje es muy útil dentro de los circuitos electrónicos debido a que los elementos dentro de él funcionan de mejor manera si el voltaje es estable. La batería provee este tipo de voltaje, y conservan esta cualidad hasta el final de su vida útil cuando finalmente se descargan.

Figura No 18. Baterías[10]

[9] Voltaje AC. Disponible en: Tomado de: http://www.matematicasypoesia.com.es/monografias/voltimetro.jpg

[10] Baterías. Disponible en: http://ntic.uson.mx/wikiseguridad/images/9/95/Baterias.jpg

Unidad de medición

Su unidad de medición es el voltio, representado por la letra (V).

Figura No 19. Diferencia potencial.[11]

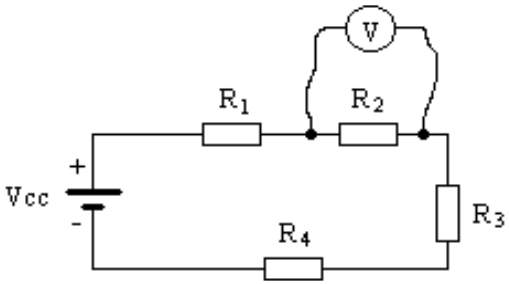

Figura No 20. Medición de voltaje de los circuitos.

Nota: en la medición de voltaje de los circuitos, la medición debe hacerse manteniendo contacto o conectando los terminales del elemento medidor (voltímetro) y el elemento a medir (generalmente una resistencia) entre sí de forma paralela como lo indica la figura anterior.

[11] Unidad de Medición. Disponible en: http://www.matematicasypoesia.com.es/monografias/voltimetro.jpg

Resistencia eléctrica

Resistencia eléctrica es toda oposición que encuentra la corriente a su paso por un circuito eléctrico cerrado, atenuando o frenando el libre flujo de circulación de las cargas eléctricas o electrones. Cualquier dispositivo o consumidor conectado a un circuito eléctrico representa en sí una carga, resistencia u obstáculo para la circulación de la corriente eléctrica[12].

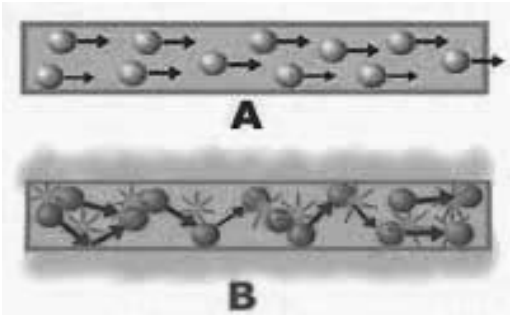

Figura No 21. Medición de voltaje de los circuitos[13]

Bit – Byte – Kilobyte

Un sistema de numeración[14] es un conjunto de símbolos y reglas que permiten representar datos numéricos. Los sistemas de numeración actuales son sistemas posicionales, que se caracterizan porque un símbolo tiene distinto valor según la posición que ocupa en la cifra.

[12] ¿Qué es la resistencia eléctrica? José Antonio E. García Álvarez. Disponible en: http://www.asifunciona.com/electrotecnia/ke_resistencia/ke_resistencia_1.htm

[13] Ibid

[14] Sistemas de Numeración. Luis González. I.E.S. Santa Eugenia. Disponible en : http://platea.pntic.mec.es/~lgonzale/tic/binarios/numeracion.html

Bit

Bit[15] es el acrónimo de "Binary digit" (dígito binario). Un bit es un dígito del sistema de numeración binario.

Mientras que en el sistema de numeración decimal se usan diez dígitos, en el binario se usan sólo dos dígitos, el 0 y el 1. Un bit o dígito binario puede representar uno de esos dos valores, 0 ó 1.

Se puede imaginar un bit como especie de un bombillo que está encendido:

Figura No 22[16] . Encendido.

O apagado:

Figura No 23[17]. Apagado.

[15] Definición de Bit. Disponible en: http://es.wikipedia.org/wiki/Bit

[16] Bombillo Encendido. Disponible en: elpolvorin.over-blog.es

[17] Bombillo Apagado. Disponible en: fundaroraima.blogspot.org

Figura No 24[18]. Sistema binario.

El bit es la unidad mínima de información empleada en informática, en cualquier dispositivo digital o en la teoría de la información. Con él, podemos representar dos valores cuales quiera, como verdadero o falso, abierto o cerrado, blanco o negro, norte o sur, masculino o femenino, rojo o azul, etc. Basta con asignar uno de esos valores al estado de "bajo" (0), y el otro al estado de "alto" (1).

Las combinaciones de bits se utilizan para representar o codificar más información en un dispositivo digital, por ejemplo, si utilizamos dos bits obtendremos:

Tabla No 2.
Configuración de bits.

0-0	Los dos están "apagados"
0-1	El primero (de derecha a izquierda) está "encendido" y el segundo "apagado"
1-0	El primero (de derecha a izquierda) está "apagado" y el segundo "encendido"
1-1	Los dos están "encendidos"

[18] Sistema Binario. Disponible en:Otroblogmas.com

Conversión binaria a decimal

Como se dijo anteriormente, los bits son dígitos del sistema de numeración binaria, eso quiere decir que se rige bajo la norma que dice que el valor de la combinación de bits es la suma de los productos de los dígitos por una base 2 elevada a un exponente desde 0, siempre de derecha a izquierda.

Ejemplo: 101101

Esto equivale a: $1*2^0 + 0*2^1 + 1*2^2 + 1*2^3 + 0*2^4 + 1*2^5$

$1 + 0 + 4 + 8 + 0 + 32$

101101 equivale a 45 en decimal.

Byte

Byte[19] es una palabra aceptada como equivalente a octeto (es decir, a ocho bits), para fines correctos, un byte debe ser considerado como una secuencia de bits contiguos, cuyo tamaño depende del código de información o código de caracteres en que sea definido.

Se usa comúnmente como unidad básica de almacenamiento de datos en combinación con los prefijos de cantidad.

Figura No 25[20]. Byte.

[19] Definición de Byte. Disponible en: http://es.wikipedia.org/wiki/Byte

[20] Definición de Byte. Disponible en: http://es.wikipedia.org/wiki/Byte

El byte es equivalente a un único caracter, como puede ser una letra, un número o un signo de puntuación.

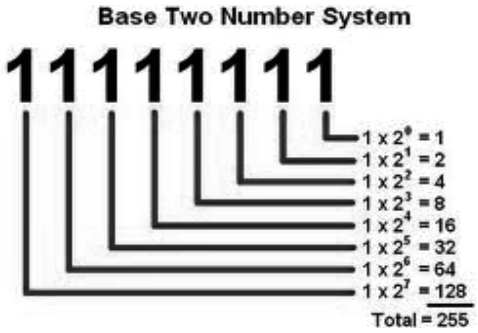

Figura No 26. Conversión de binario a decimal.

Un byte en Arduino almacena un número si signo de 8 bits desde 0 hasta 255.

Ejemplo: Byte a = B10010

Donde B es binario y 10010 equivale a 18 en decimal.

KiloByte

Figura No 27. KiloByte.

Un Kilobyte[21] (abreviado como KB o Kbyte) es una unidad de medida equivalente a mil bytes de memoria de ordenador o de capacidad de disco. Por ejemplo, un dispositivo que tiene 256K de memoria puede almacenar aproximadamente 256.000 bytes (o caracteres) de una vez.

En sistemas decimales, kilo significa 1.000, pero el mundo de los ordenadores se basa en un sistema binario de dos en vez de diez. Así pues, un kilobyte es realmente 1.024 (210) bytes. Para distinguir entre una K decimal (1.000) y una K binaria (1.024), el IEEE ha sugerido usar una k minúscula para un kilo decimal y una K mayúscula para un kilo binario.

Nota: Kb es el kilobit y KB es kilobyte.

Conociendo Arduino
Introducción a microcontroladores

¿Qué es un micro controlador?

Figura No 28[22]. Microcontrolador

[21] Definición de kilobyte. Disponible en: http://www.masadelante.com/faqs/kilobyte

[22] MicroControlador. Disponible en: memito-trabajospics.blogspot.com

Un microcontrolador[23] es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, memoria y unidades de entrada/salida, es decir, se trata de un computador completo en un solo circuito integrado.

Ventajas de un microcontrolador

Figura No 29[24]. Microcontrolador dsPIC

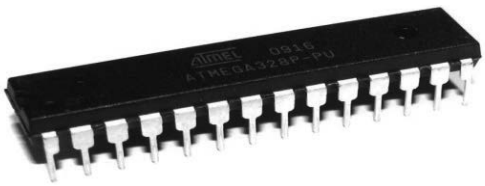

Figura No 30[25]. Microcontrolador.

Los microcontroladores son diseñados para disminuir el coste económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la CPU, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación.

[23] MicroControlador. Disponible en: http://microcontroladores-e.galeon.com/

[24] MicroControlador. Disponible en: http://microcontroladores-e.galeon.com/

[25] Arduino. Disponible en: proyectosfie.com

¿Qué es arduino?

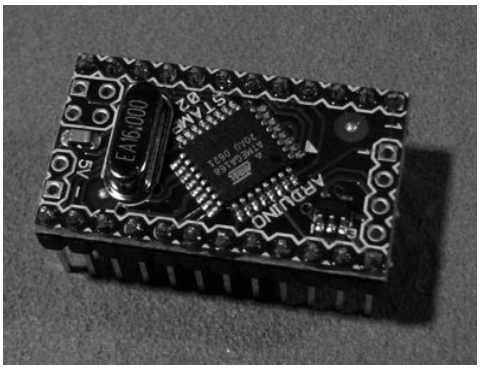

Figura No 31[26]. Arduino

Arduino[27] es una plataforma de electrónica abierta para la creación de prototipos basada en software y hardware flexibles y fáciles de usar. Se creó para artistas, diseñadores, aficionados y cualquier interesado en crear entornos u objetos interactivos.

Figura No 32[28]. Ventajas de Arduino.

[26] Placa Arduino. Disponible en:elecronicaestudio.com

[27] Placa Arduino Diecimilla. http://www.arduino.cc/es/.

[28] Imagen tomada de antoniotoriz.blogspot.com

Ventajas de Arduino

1. Lo pueden utilizar artistas, diseñadores, aficionados o cualquier otro interesado.
2. Facilidad de instalación tanto de sus componentes como de su software.
3. Se encuentran gran cantidad de proyectos en la comunidad web.
4. Es abierto, lo cual nos permite modificar con libertad.
5. Sus componentes son reutilizables.

Modelos de placas Arduino

Arduino Diecimila

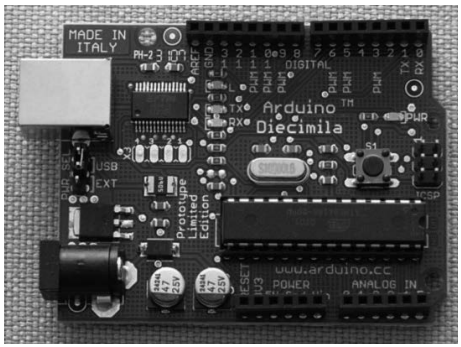

Figura No 33[29]. Arduino Diecimilla.

La Arduino Diecimila[30] es una placa microcontroladora basada en el chip ATmega168. Tiene 14 entradas/salidas digitales, 6 entradas analógicas, un cristal de 16MHz, conexión USB y botón de reseteo. Contiene todo lo necesario para el soporte del microcontrolador.

[29] Arduino Diecimilla. Disponible en: ingeniosolido.com

[30] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >

Arduino Uno

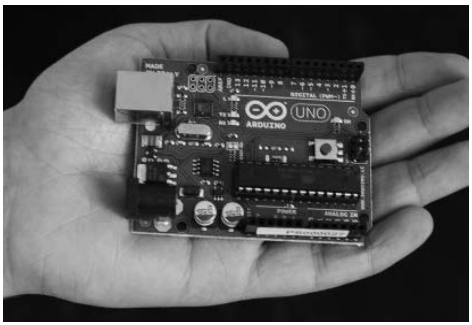

Figura No 34[31]. Arduino Uno.

El Arduino Uno[32] es una placa electrónica basada en el ATmega328. Tiene 14 entradas/salidas digitales pines, 6 entradas analógicas, un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reseteo.

Arduino Mega

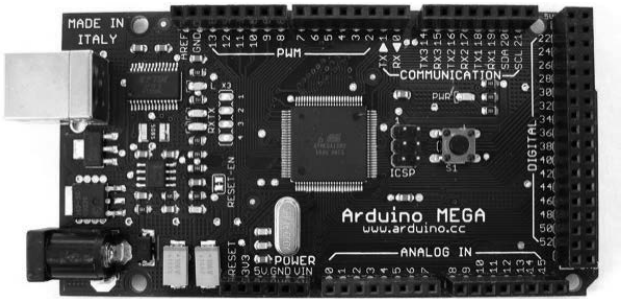

Figura No 35[33] Arduino Mega.

[31] Arduino Uno. Disponible en: www.arduino.cc

[32] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >

[33] Arduino Mega. Disponible en: www.arduino.cc

El Arduino Mega[34] es una placa microcontrolador basada en ATmeg1280. Tiene 54 entradas/salidas digitales, 16 entradas analógicas, 4 UARTS (puertos serie por hardware), un cristal oscilador de 16MHz, conexión USB, entrada de corriente, conector ICSP y botón de reseteo.

Arduino Mega2560

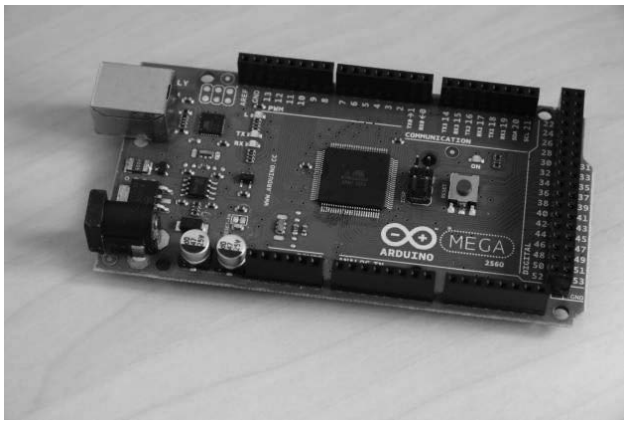

Figura No 36[35] Arduino Mega2560.

El Arduino Mega2560[36] es una placa electrónica basada en el Atmega2560. Cuenta con 54 pines de entradas/salidas digitales, 16 entradas analógicas, 4 UART (puerto serie del hardware), un oscilador de cristal de 16 MHz, una conexión USB, un conector de alimentación, una cabecera ICSP y un botón de reseteo.

[34] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >

[35] Arduino Mega 2560. Disponible en: chemicaloliver.net

[36] ©Arduino Octubre 3 2011< http://www.arduino.cc/es/ >

Arduino Fio

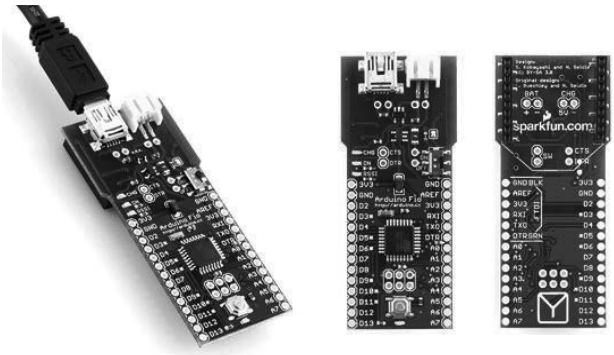

Figura No 37[37] Arduino FIO.

El Arduino Fio[38] es una placa para microcontrolador basada en el ATmega328P. Funciona a 3.3V y 8MHz. Tiene 14 pines de entrada/salida digitales, 8 entradas analógicas, un resonador en placa, un botón de reinicio (reseteo) y agujeros para montar conectores de pines.

Tiene conexiones para una batería de polímero de litio e incluye un circuito de carga a través de USB. En el reverso de la placa tiene disponible un zócalo para módulos XBee (módulos de transmisión inalámbrica), es decir, el Arduino FIO está diseñado para aplicaciones inalámbricas. El usuario puede subir sus sketches (bocetos) con un cable FTDI (Futura Tecnología de Dispositivos Internacionales) o una placa adicional adaptadora. Además, si utiliza un adaptador de USB a XBee modificado, como el USB Explorador de XBee, el usuario puede subir sketches de forma inalámbrica. La tarjeta viene sin conectores pre-montados, permitiendo el uso de diversos tipos de conectores o la soldadura directa de los cables.

[37] Arduino FIO. Disponible en: www.arduino.cc

[38] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >

Arduino Nano

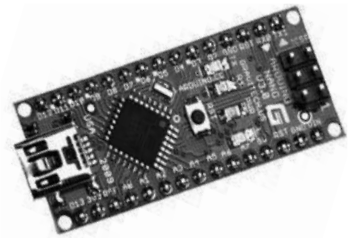

Figura No 38[39]. Arduino Nano.

El Arduino Nano[40] es una pequeña y completa placa basada en el ATmega328 (Arduino Nano 3.0) o ATmega168 (Arduino Nano 2.x) que se usa conectándola a una protoboard (placa de prueba). Tiene más o menos la misma funcionalidad que el Arduino Duemilanove, pero con una presentación diferente. No posee conector para alimentación externa, y funciona con un cable USB Mini-B en vez del cable estándar.

Arduino Lilypad

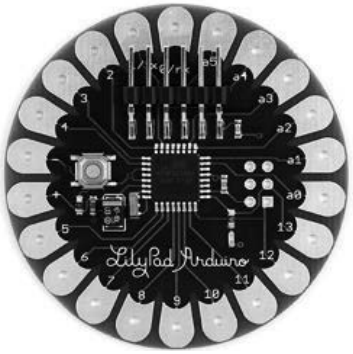

Figura No 39[41]. Arduino Lilypad.

[39] Arduino NANO. Disponible en: www.robotshop.com

[40] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >

[41] Arduino Lillypad. Disponible en: www.arduino.cc

El LilyPad Arduino[42] es una placa con microcontrolador diseñado para prendas y e-textiles. Se puede utilizar con complementos similares como fuentes de alimentación y sensores actuadores unidos por hilo conductor. La placa está basada en el ARmega168V (la versión de bajo consumo del ATmega168).

Arduino Duemilanove

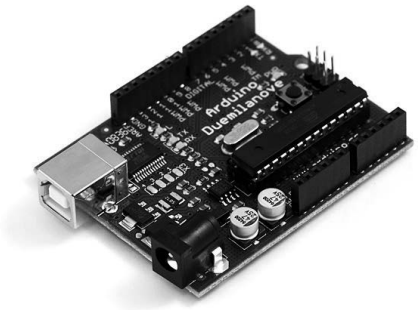

Figura No 40[43]. Arduino Duemillanove.

El Arduino Duemilanove[44] ("2009") es una placa con microcontrolador basada en el ATmega168 ó el ATmega328. Tiene 14 pines con entradas/salidas digitales, 6 entradas analógicas, un cristal oscilador a 16Mhz, conexión USB, entrada de alimentación, una cabecera ISCP y un botón de reseteo.

El Arduino Duemilanove puede ser alimentado vía conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.

[42] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >

[43] Arduino Duemillanove. Disponible en: olimex.cl

[44] ©Arduino Octubre 4 2011< http://www.arduino.cc/es/ >

¿Qué es un diodo?

Diodo

Figura No 41[45]. Diodo.

Es un componente electrónico que permite el paso de la corriente en un solo sentido.

Partes del diodo

Está compuesto por dos terminales polarizadas, una terminal positiva (ánodo) y una terminal negativa (cátodo).

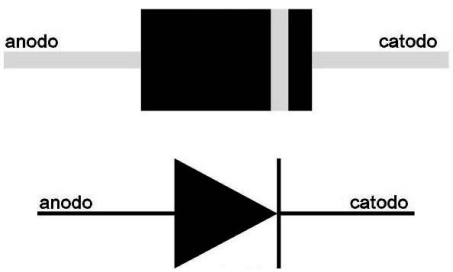

Figura No 42[46]. Partes del diodo 1.

[45] Diodo. Disponible en: http://misresidencias.webnode.com/album/fotogaleria/diodo-1n4001-jpg/

[46] Partes del diodo. Disponible en: http://www.olimex.cl/present.php?page=tutorial-diodos

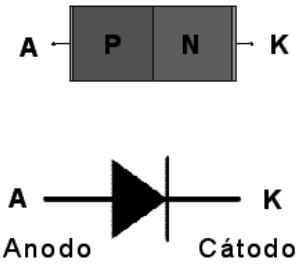

Figura No 43[47]. Partes del diodo 2.

Funcionamiento

Permiten la conducción de corriente cuando están conectados en polarización directa, es decir, cuando al terminal positivo del diodo se conecta a un polo positivo y el terminal negativo a tierra o un voltaje negativo.

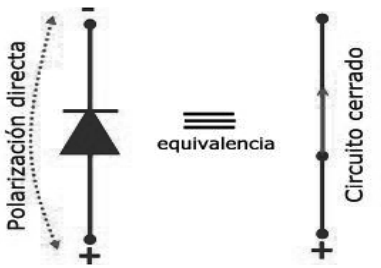

Figura No 44[48]. Funcionamiento del diodo.

Características

Entre muchas de las características técnicas que posee el diodo como cualquier componente electrónico, se debe tener encuenta primeramente la capacidad de corriente que éste soporta. Generalmente, entre mayor sea el tamaño del diodo, mayor cantidad de corriente soporta.

[47] Partes del diodo. Disponible en: http://www.ucontrol.com.ar/wiki/index.php?title=Diodo

[48] Funcionamiento del diodo. Disponible en: http://www.ifent.org/lecciones/diodo/equivalente.asp

Figura No 45[49]. Características del diodo.

Usos comunes

El uso más común se da en las fuentes de voltaje DC -en las etapas de rectificado-, ya que permiten pasar solamente los ciclos positivos de la onda de voltaje AC.

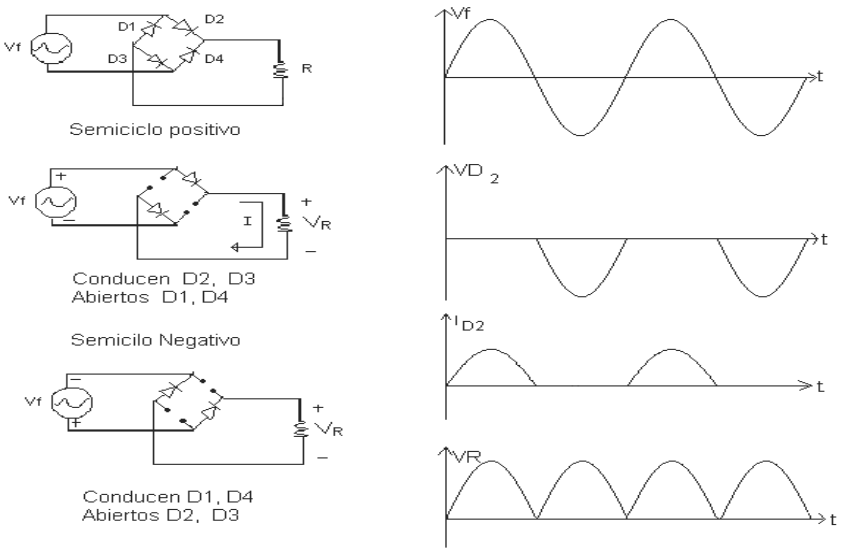

Figura No 46[50]. Usos del diodo.

[49] Características del diodo. Disponible en: http://www.electrozone.com.mx/tblarticulos_list.php?goto=29

[50] Usos del diodo. Disponible en: http://teleco10-11.blogspot.com/2010/06/macce.html

Puente rectificador

Con el fin de no utilizar los 4 diodos rectificadores existen componentes en el mercado que encapsulan estos cuatro diodos en un solo dispositivo.

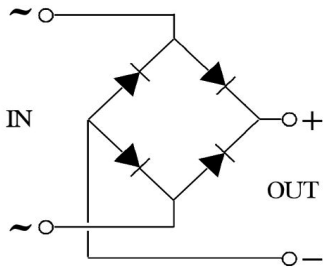

Figura No 47[51]. Puente rectificador.

Figura No 48[52]. Puente rectificador.

Diodos Zener

Estos diodos, por lo general, cumplen su función polarizados inversamente y sirven de reguladores de tensión en partes de circuitos en donde esto se disponga. Para lograr su fusión de regulador de voltaje, éstos absorben la corriente que sea necesaria para poder mantener la tensión. El terminal más cercano a la raya negra que está sobre el componente, es la termina negativa.

[51] Puente rectificador: http://www.pisotones.com/GF/GF.htm

[52] Puente rectificador. Disponible en: http://www.promelsa.com.pe/productos_list.asp?id_linea=014&id_sublinea=1&id_familia=05&saldos=&pm_list=L

Figura No 49[53]. El diodo y sus componentes.

Otros Tipos de Diodos

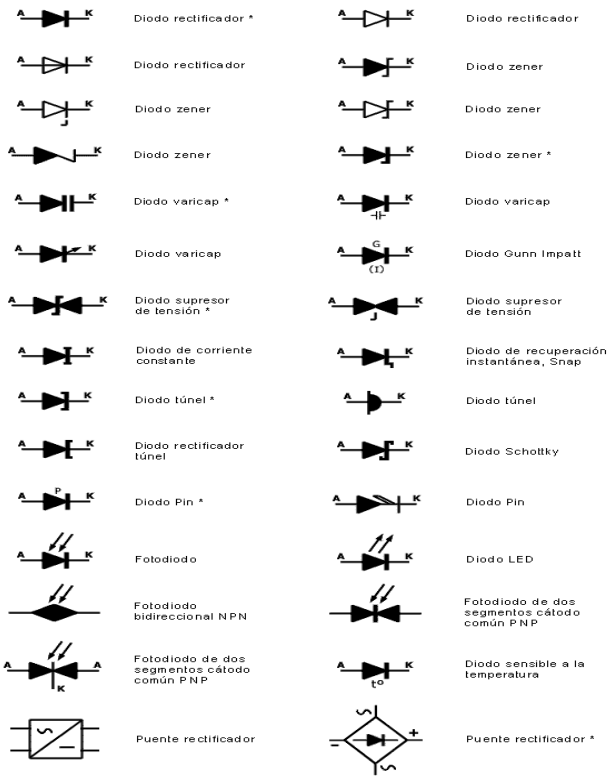

Figura No 50[54]. Símbolos representativos de los diodos.

[53] Componentes del diodo. Disponible en: http://www.store.reparacionlcd.com/DIODO%20ZENER%20Z.8V2

[54] Símbolos representativos de los diodos. Disponible en: http://www.geocities.ws/jjrc_79/electronica/fundamentos/semiconductores/semiconductores.htm

Práctica diodo rectificador

El diodo más común es el 1N4001. Comprobaremos entonces el efecto rectificación, en el cual, sólo conduce corriente cuando está polarizado directamente.

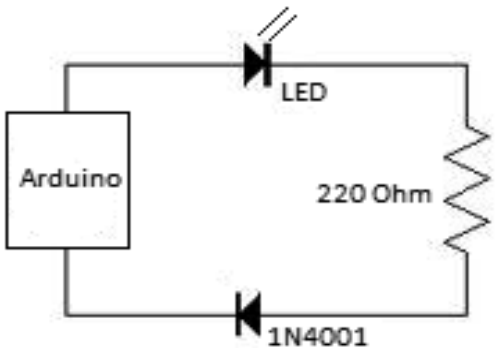

Figura No 51. Práctica de diodo Parte 1.

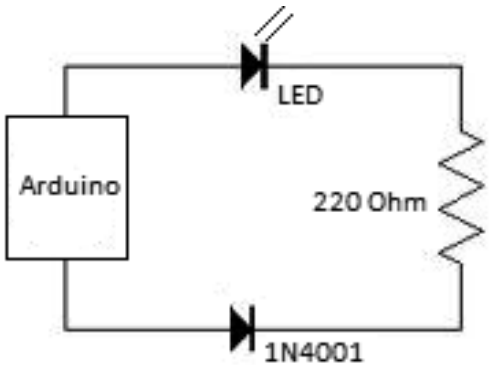

Figura No 52. Práctica de diodo Parte 2.

Fotoresistores

Figura No 53[55]. Fotoresistores.

¿Qué es un fotoresistor?

Es un elemento electrónico que permite la variación de su resistencia interna mediante cierta excitación lumínica.

Figura No 54[56]. Ejemplos de fotoresistores.

[55] Fotoresistores. Disponible en: http://upload.wikimedia.org/wikipedia/commons/3/32/LDR.jpg

[56] Ejemplos de fotoresistores. Disponibles en: http://3.bp.blogspot.com/_N-1jF8SnkhM/Sz1pAY4CmpI/AAAAAAAAAA4/CtQ3Nj85h5I/s320/schematics_photoresistors.jpg

Partes de un fotoresistor

Un fotoresistor se encuentra conformado por su célula o celda y sus dos terminales.

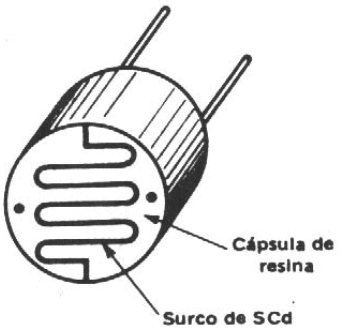

Figura No 55[57]. Ejemplos de fotoresistores 1.

Su símbolo eléctrico es el siguiente:

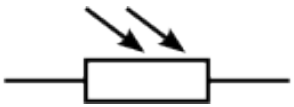

Figura No 56. Ejemplos de Fotoresistores 2

Funcionamiento

Su funcionamiento radica en que su composición semiconductora posee ciertas características que permiten que en el momento de incidir luz de alta frecuencia sobre éste, los fotones sean absorbidos por la elasticidad del semiconductor, permitiendo de esta manera, dar suficiente energía a los electrones para saltar a la banda conductora. Este fenómeno del movimiento de los electrones conduce la electricidad de cierta manera que provoca una reducción en la resistencia del elemento.

[57] Ejemplo de Fotoresistores. Disponible en: http://nottheremin.files.wordpress.com/2009/02/fotoresistencia.gif?w=510

Figura No 57[58]. Funcionamiento de fotoresistores

Características

Como característica principal se encuentra que entre más incidencia de luz exista sobre el elemento, menos será su resistencia eléctrica. Por otro lado, posee cierto retardo en la variación de su resistencia, razón por la cual, es menos efectiva en este aspecto para el control de un circuito en comparación con otros dispositivos.

Figura No 58[59]. Funcionamiento de fotoresistores.

[58] Funcionamiento de los fotoresistores. Disponible en: http://www.cienciafacil.com/cds_photocell.jpg

[59] Funcionamiento de los Fotoresistores. Disponible en: http://electronica.ugr.es/~amroldan/asignaturas/curso04-05/cce/prac1/ldr.gif

Tipos de fotoresistores

Se fabrican según las diferentes necesidades del mercado; pueden ser para relojes con radio, cámaras, alarmas de seguridad, alumbrado público, medidores de luz, sistemas contraincendios y para gamas bajas de radiación infrarroja se pueden encontrar como fotoconductores de Ge:Cu.

Figura No 59[60]. Funcionamiento de fotoresistores

Prácticas con fotoresistores

A una de las salidas digitales del Arduino, conectaremos un LED en serie con una resistencia de 220 Ohm y la fotoresistencia. Notaremos que al incidir luz sobre la fotoresistencia aumentará la luminosidad del LED, ya que hay menor resistencia y fluir a mayor corriente. Y hará el efecto contrario al no incidir luz sobre ésta.

[60] Funcionamiento de los Fotoresistores. Disponible en: http://repositorio2.masoportunidades.com.ar/ARG01/103/28/10144541_2_18/fotos/2_10144541_2_2006726-17-4-47.jpg

Hardware y software libre

Hardware

Figura No 60[61]. Hardware de computadores.

El Hardware es denominado como la parte física o tangible, incluyendo cualquier otro elemento físico involucrado.

Software

Figura No 61[62]. Software de computadores.

[61] Hardware de Computadores. Disponible en: pcexpertos.com

[62] Software de Computadores. Disponible en: bibliotecadeinvestigaciones.wordpress.com

El software se refiere a la parte intangible, aquella que no podemos tocar, como datos almacenados y programas que tal vez utilizamos a diario en un computador; por ejemplo: utilizamos el programa Microsoft Office Word a la hora de redactar una carta, un memorando o una hoja de vida.

Hardware libre

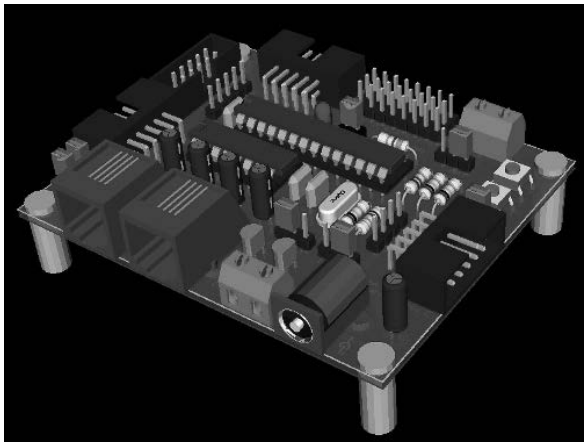

Figura No 62[63]. Hardware libre.

El Hardware libre es aquella parte física o dispositivo creado de forma abierta con el objetivo principal de que todas las personas puedan acceder a su plano de construcción y a las partes con las que fue construido, buscando así que en su utilización se puedan realizar cambios, corrección de errores y mejoras en su diseño.

Un vivo ejemplo de hardware libre es la placa Arduino de la cual ya hemos hablado y seguiremos haciéndolo más adelante.

Otro éxito ha sido Ronja (Reasonable Optical Near Joint Access), lo cual significa, Acceso Óptico Razonable de Nodo Cercano.

[63] Hardware Libre. Disponible en: iearobotics.com

Figura No 63[64]. Ronja.

Fue creado con el objetivo de establecer enlaces de datos de transmisión óptica punto a punto inalámbrica (óptica de espacio libre); es excelente para ser trabajado por inexpertos ya que las instrucciones de construcción están escritas por y para una persona sin experiencia, explicando las operaciones básicas de taladrado, soldadura, etc. Se muestran varias técnicas para reducir al mínimo los errores en lugares críticos y acelerar el trabajo - plantillas de taladrado, revisión de las soldaduras, procedimientos de prueba, etc. Es posible descargar los diseños de circuitos impresos listos para su construcción con todas las instrucciones.

Software libre

Figura No 64[65]. Software libre.

[64] Ronja. Disponible en: blog.marques.cx

[65] Software libre. Disponible en: tucamon.es

El software libre es aquel al que se le conoce su código fuente para que el usuario pueda emplearlo de la manera que más le parezca, además de poder modificarlo, distribuirlo y adaptarlo a sus necesidades.

Seguramente hemos escuchado el nombre de muchos de estos tipos de software como Linux, Mozilla Firefox y Ubuntu, pero sencillamente, tal vez no sabíamos que eran libres.

Hay muchos productos de software libre que tienen mucha acogida, como por ejemplo, el navegador Mozilla Firefox, que es el segundo navegador de internet más utilizado por los usuarios. Éste tiene características muy notables que hacen que sea realmente exitoso en diversos aspectos; lo mejor de todo es que cada uno de nosotros podemos ser parte de estos proyectos, así como también generar y ser emprendedores de nuestros propios proyectos de software libre.

Figura No 65[66]. Browser Firefox.

[66] Browser Firefox. Disponible en: fondosypantallas.com

Instalación de Arduino

Para empezar deberás tener a tu alcance una placa Arduino UNO (opcional -el modelo de la placa es variable) y un cable USB como el que se usa para conectar una impresora USB.

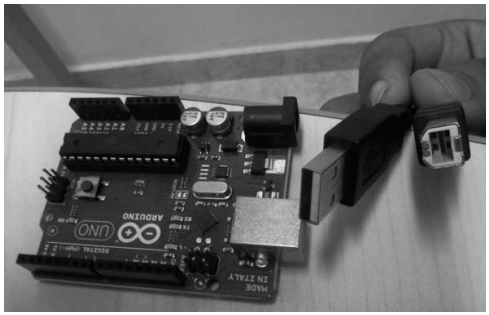

Figura No 66. Instalación con Arduino.

Descargamos el IDE de Arduino dependiendo del modelo de la placa, en este caso, para la placa Arduino UNO descargaremos Arduino-0022, pero puede ser otra versión Arduino-00XX que incluya los drivers de instalación de la placa.

Cuando la descarga finalice se descomprime el fichero, tratando siempre de mantener la estructura de los archivos, como muestra la imagen:

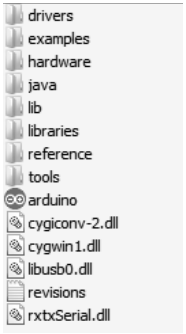

Figura No 67. Directorio de Arduino.

Conecta la placa al ordenador y un LED verde se encenderá indicando la alimentación (mientras la placa se encuentre conectada, el LED deberá permanecer encendido).

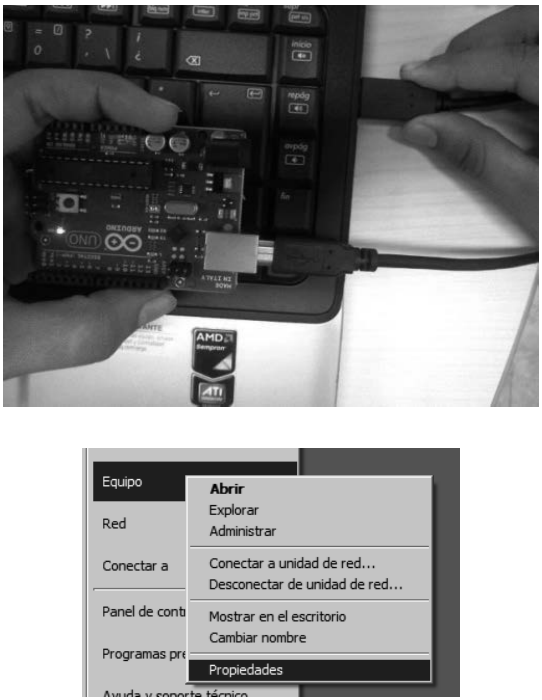

Figura No 68. Instalación de la placa Arduino al PC.

Tu sistema operativo te indicará inicializar la instalación de los drivers (siempre y cuando nunca se haya trabajado con la placa) y deberás indicarle que no a todas las opciones. Luego debes entrar a las propiedades del equipo (Mi PC), das click derecho sobre él y listo. En las propiedades se entrará al administrador de dispositivos; una vez aquí, encuentras todos los dispositivos que están instalados en el equipo, también se puede actualizar su software controlador, comprobar si algún dispositivo funciona correctamente y modificar sus opciones de configuración.

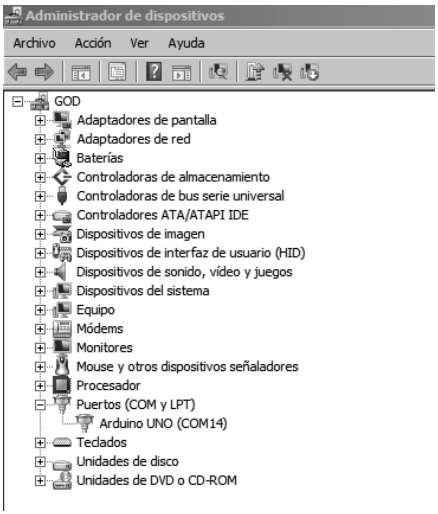

Figura No 69. Configuración de Arduino al PC.

En la opción Puertos (COM y LPT) se encuentra ubicada nuestra placa, debemos tener en cuenta en qué número de puerto se encuentra, ya que puede variar su ubicación (en la imagen vemos que se encuentra en el puerto número 14).

Le damos click derecho donde se encuentra el nombre de nuestra placa y escogemos la opción actualizar software controlador, nos aparecerá una pantalla como la siguiente:

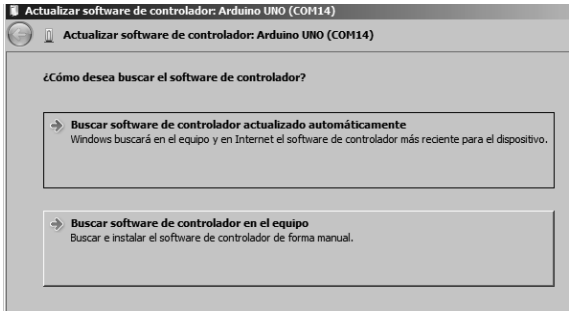

Figura No 70. Actualización de controlador 1.

Y escogemos la opción “Buscar software de controlador en el equipo”.

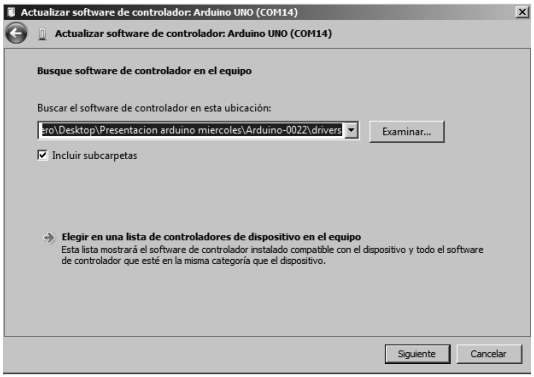

Figura No 71. Actualización de controlador 2

Le damos examinar y buscamos la carpeta drivers que se encuentra en nuestra carpeta IDE Arduino-00XX que habíamos descargado antes e incluimos las subcarpetas, luego le damos siguiente. Al finalizar nos dirá que nuestro software está actualizado y procedemos a cerrar.

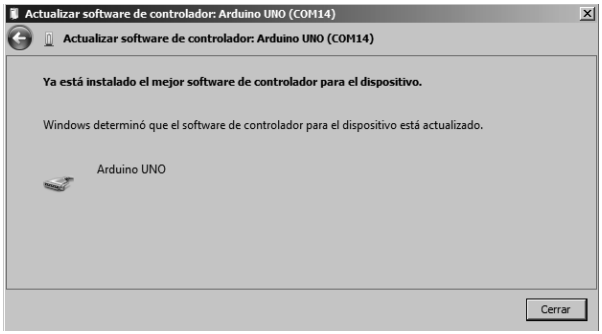

Figura No 72. Actualización de controlador 3.

Nuestra placa está instalada, para comenzar a trabajar con ella debemos ejecutar nuestra aplicación Arduino que se encuentra en nuestra carpeta Arduino-00XX.

Figura No 73. Ícono de Arduino.

Se abrirá una ventana como en la siguiente imagen:

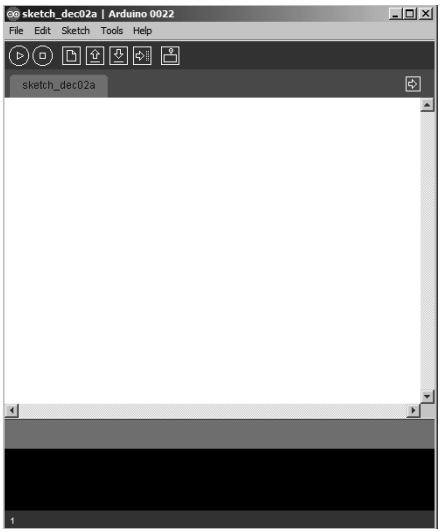

Figura No 74. Software Arduino.

En la cinta de opciones seleccionamos Tools, entramos a la opción Board y escogemos la referencia de nuestra placa.

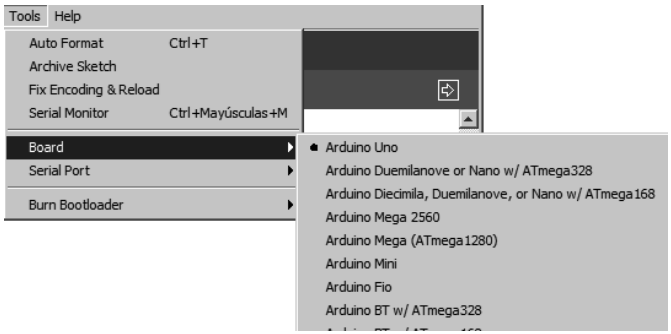

Figura No 75. Opciones de Tools software Arduino.

Luego en Tools seleccionamos Serial Port (puerto serial) y escogemos el puerto en el que se encuentra la placa; este puerto lo vimos en el administrador de dispositivos cuando actualizamos el software de controlador.

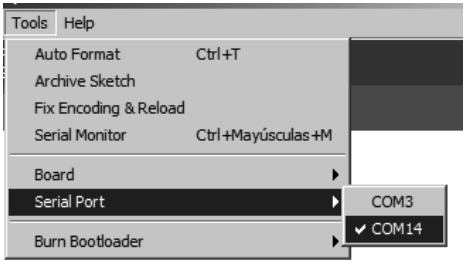

Figura No 76. Opciones de Serial Port software Arduino.

Nuestro IDE reconoce nuestra placa y vamos a probarlo con un ejemplo sencillo:

Nuestra placa tiene un LED incorporado en el pin 13, haremos que este LED encienda desde nuestro IDE.

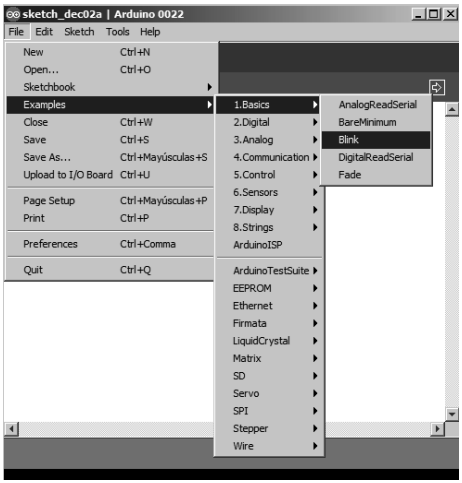

Figura No 77. Ejemplos cargados en la plataforma.

En nuestra cinta de opciones entramos a File, luego a Examples, después pasamos a Basics y le damos BLink. Así, nos abrirá este sencillo programa, que hace que el LED ubicado en nuestro pin 13 parpadee.

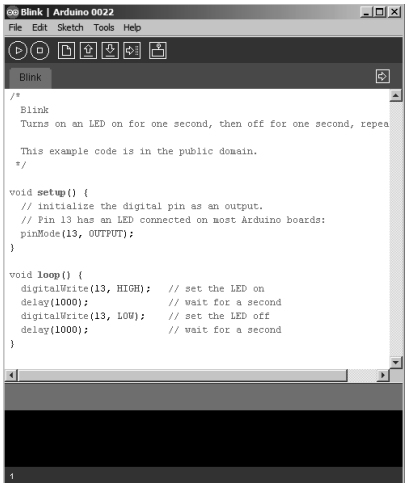

Figura No 78. Ejemplos código fuente plataforma.

En nuestra barra:

Figura No 79. Barra Software Arduino.

Presionamos

Figura No 80. Opción Upload Software Arduino.

El cual sube el programa a la placa, y en la consola debe salir:

Figura No 81. Realizando Upload en Board.

Y al finalizar, si todo es correcto debe aparecer:

Figura No 82. Indicación de NO errores.

Indicando que no hay errores. Asimismo, miramos nuestra placa y nos damos cuenta que el LED que se encuentra en el pin 13 parpadea. Entonces nuestra placa esta lista y funcionando.

Interruptores

Figura No 83[67]. Tipo de interruptor 1.

Figura No 84[68]. Tipo de interruptor 2.

¿Qué es un interruptor?

También llamado switch, es un dispositivo diseñado de tal forma que permita conducir e interrumpir el paso de la corriente, sus variaciones y combinaciones permiten incluso el desvío de la corriente en casos necesarios.

La forma más básica de un interruptor es la siguiente:

Figura No 85[69]. Interruptor pulsante.

[67] Tipos de Interruptor. Disponible en: http://parts.digikey.com/1/parts/978941-switch-tact-6mm-momentary-250gf-tl1105af250q.html

[68] Tipo de Interruptor. Disponible en: http://andreagarciapqpi.wordpress.com/2011/09/29/elementos-basicos-de-electricidad-y-electronica/

[69] Imagen tomada de http://www.incopia2.com/shop/interruptor-basculante-empotrable-p-6133.html

El símbolo e un interruptor básico es el siguiente:

Figura No 86. Símbolo de interruptor básico.

Generalidades

Normalmente abierto (NA). Existen interruptores con un diseño de fábrica para que en sus estados normales, no conduzcan corriente, a menos que el usuario u otro mecanismo así lo deseen.

Normalmente cerrado (NC). Configuración de los interruptores con la cual, en sus estados normales, permite siempre el paso de la corriente, siendo interrumpida por acción del usuario u otro mecanismo. Esta configuración se da comúnmente en los interruptores de tipo pulsador.

Polos. Es la cantidad de conexiones individuales que permite manejar un interruptor. Ejemplo, la acción sobre un interruptor de dos estados (on-off) de dos polos podría permitir la activación de dos dispositivos al mismo tiempo, controlando el paso de corriente hacia cada dispositivo a través de conexiones internas individuales.

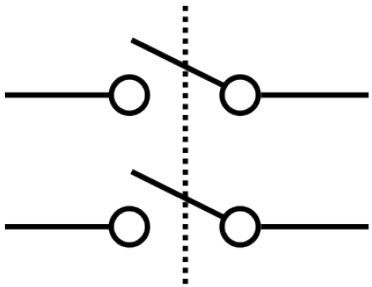

Figura No 87. Interruptores de dos polos una vía.

Vías (Tiros). Es la cantidad de conexiones a las que se le puede dar paso de corriente de una sola fuente. Un ejemplo: un interruptor de dos polos y tres v**ías** podría fácilmente controlar el encendido y apagado de las luces de dos semáforos al mismo tiempo.

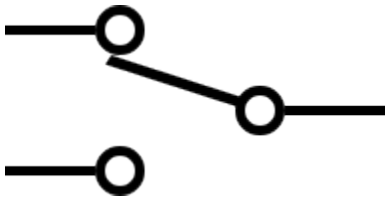

Figura No 88. Interruptores de un polo dos vías.

Interruptor basculante

Su mecanismo de acción consiste en una palanca que debe inclinarse de un lado al otro para cambiar el estado del interruptor.

Figura No 89[70]. Interruptor basculante.

[70] Interruptor basculante. Disponible en: http://www.elotrolado.net/hilo_tutorial-dump-volcado-de-los-nuevos-liteon-flasheo_1338376_s520

Botón o Pulsador

Dispositivo interruptor que funciona cortando o permitiendo el flujo de la corriente según esté o no presionado, es decir actúa por instantes de tiempo.

Figura No 90[71]. Pulsador.

Funcionamiento y forma de pulsadores

Figura No 91[72].
Funcionamiento pulsador 1.

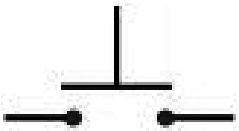

Figura No 92[73].
Funcionamiento pulsador 2.

[71] Pulsador. Disponible en: http://www.shoptronica.com/817-pulsadores-de-panel-de-21x10mm.html

[72] Funcionamiento Pulsador. Disponible en: http://mundoelectronics.blogspot.com/2009/04/el-plc-diagramas-de-escalera-i.html

[73] Funcionamiento Pulsador. Disponible en: http://mundoelectronics.blogspot.com/2009/04/el-plc-diagramas-de-escalera-i.html

Figura No 93[74].
Pulsador 1.

Figura No 94[75].
Pulsador 2.

Figura No 95[76].
Pulsador 3.

Interruptor magnetotérmico o interruptor automático

Dispositivo utilizado para protecciones de circuitos y redes eléctricas en casos de cortocircuito o sobrecargas de corriente. Éste detecta la cantidad de corriente que pasa a través de él por medio de temperatura. Cuando la corriente es mayor de un límite establecido por la fábrica, él cambia de estado e interrumpe súbitamente el paso de corriente.

Figura No 96[77].
Interruptor 1.

Figura No 97[78].
Interruptor 2.

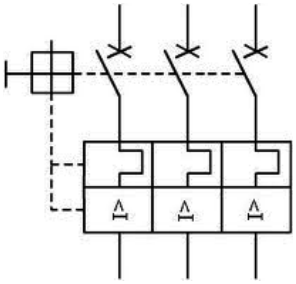

Figura No 98[79].
Interruptor 2.

[74] Tipos de Pulsador. Disponible en: http://www.shoptronica.com/703-pulsador-tact-switch-de-6x6mm-boton-de-13mm-de-alto.html

[75] Tipos de Pulsador. Disponible en:http://www.shoptronica.com/1446-pulsador-tact-switch-de-6x6mm-2pin.html

[76] Tipos de Pulsador. Disponible en:http://www.ledcontrols.com.mx/ver.php?modelo=147

[77] Tipos de interruptor. Disponible en: Imagen tomada de http://www.mercamania.es/a/listado_productos/idx/5060300/mot/Interruptor_magnetotermico_20_a/listado_productos.htm

[78] Tipos de interruptor. Disponible en: http://anuncios.ebay.es/compraventa/interruptor-magnetotermico-abb-3p-n-10a-s203/10170052

[79] Tipos de interruptor. Disponible en: http://electricidad-viatger.blogspot.com/2008/07/interruptores-automticos-magnetotrmicos.html

Reed switch

Es un interruptor encapsulado en un tubo de vidrio al vacío que cambia de estado al detectar a su alrededor un campo magnético.

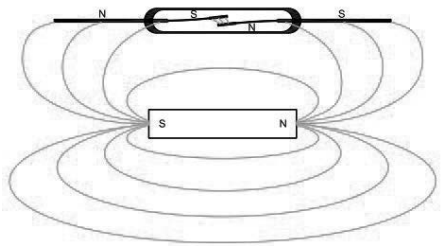

Figura No 99[81]. Switch 1.

Figura No 100[82]. Switch 2.

Dip switch

Conjunto de interruptores encapsulados en un elemento diseñado para utilizarse en placas de circuitos integrados. Es utilizado en para establecer las configuraciones en el hardware.

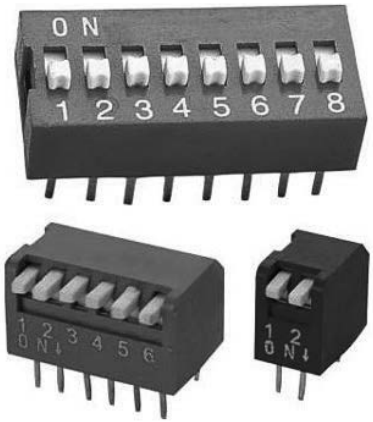

Figura No 101[82]. Dip switch.

[80] Tipos de Switch. Disponible en: http://ayudaelectronica.com/que-es-un-reed-switch/

[81] Tipos de Switch. Disponible en: http://parts.digikey.es/1/2/reed-switches

[82] Dip Switch. Disponible en: http://sigmaelectronica.net/otros-productos-c-27.html

Interruptores luminosos

Dispositivos interruptores que iluminan cuando se encuentran en determinado estado o posición.

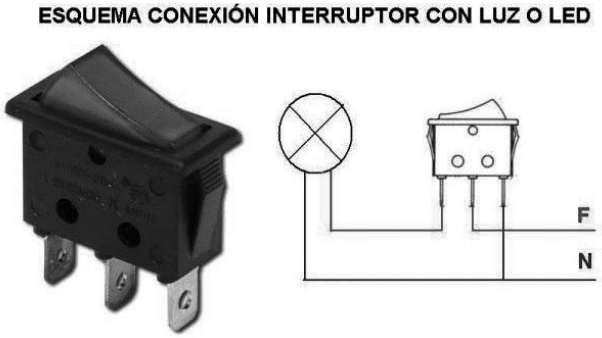

Figura No 102[83]. Esquema de conexión interruptor con luz o LED

Fusibles

Dispositivo utilizado como protección en circuitos o dispositivos en caso de que ocurra una sobrecorriente. Consta de un filamento o lámina que se desintegra con la temperatura que se produce por el mismo debido al paso de la corriente. El filamento o lámina soporta un límite de temperatura.

Figura No 103[84] Esquema de conexión interruptor con luz o LED

[83] Esquema de Conexión Interruptor con Luz. Disponible en: http://comohacerunproyectorcasero2.blogspot.com/

[84] Esquema de Conexión Interruptor con Luz. Disponible en: http://andreagarciapqpi.wordpress.com/2011/09/29/elementos-basicos-de-electricidad-y-electronica/

Interruptores fotoeléctricos

Dispositivo que funciona con tecnología de rayos infrarrojos. Tiene la capacidad de emitir una luz infrarroja y de detectar la emisión de ésta, de forma que si el haz de luz (invisible al ojo humano) es interrumpido, permitirá o no el paso de corriente en las terminales de la misma.

Figura No 104[85]. Interruptor 1.

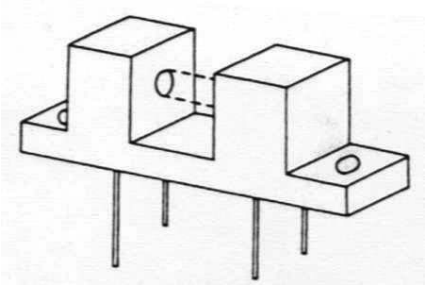

Figura No 105[86]. Interruptor 1.

[85] Tipos de Interruptores. Disponible en: http://spanish.alibaba.com/product-free/photoelectric-sensors-bs5-series-112019309.html

[86] Tipos de Interruptores. Disponible en: http://www.ucm.es/info/vivataca/anteriores/veintitres/docencia.htm

Introducción a un Simulador de Arduino

Virtual Breadboard

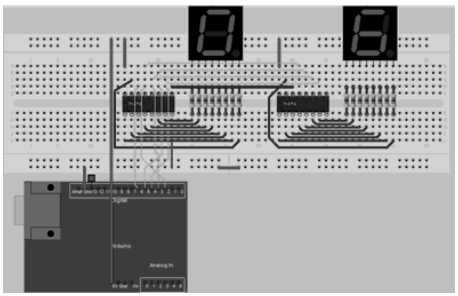

Figura No 106[87] . Simulador Arduino.

Virtual Breadboard es una simulación y entorno de desarrollo de aplicaciones integradas que utilizan los microcontroladores, es fácil de usar y puede sustituir una protoboard para experimentar con nuevos diseños.

Además, Virtual Breadboard nos ofrece:

1. Experimentar la electrónica con seguridad.
2. Simular y visualizar de forma interactiva.
3. Compartir las creaciones en vivo.
4. Aprender con ejemplos.

Antes de empezar a trabajar con este simulador, debemos instalar Microsoft Visual J# 2.0 del que podemos descargar su instalador fácilmente.

Figura No 107. Ícono instalador.

[87] Simulador Arduino. Disponible en. www.virtualbreadboard.com

Lo ejecutamos y saldrá lo siguiente:

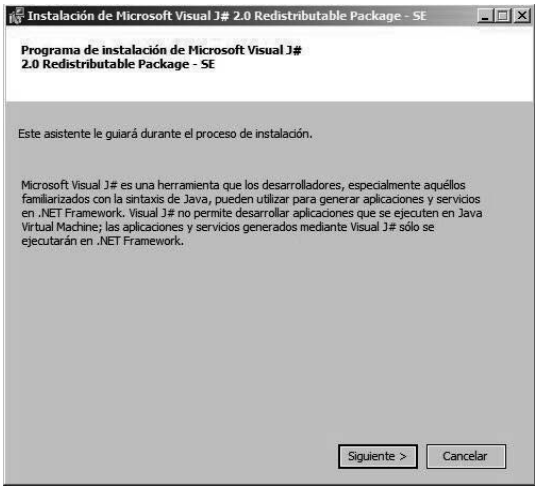

Figura No 108. Instalando Visual J# Parte 1.

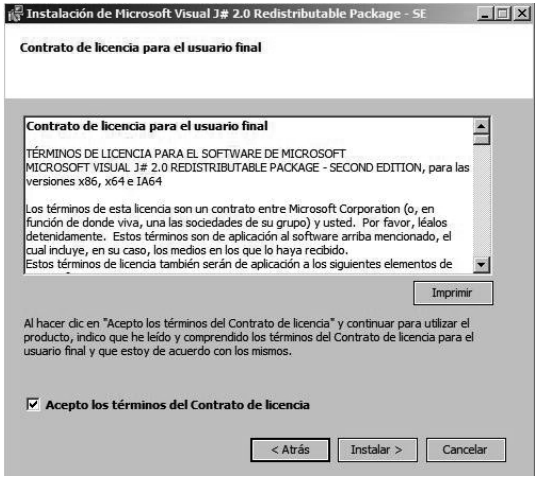

Figura No 109. Instalando Visual J# Parte 2.

Aceptamos los términos de licencia y le damos instalar. Entonces, comienza la instalación:

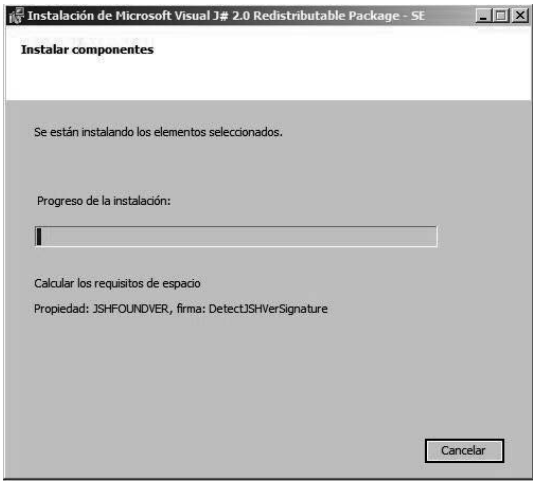

Figura No 110. Instalando Visual J# Parte 3.

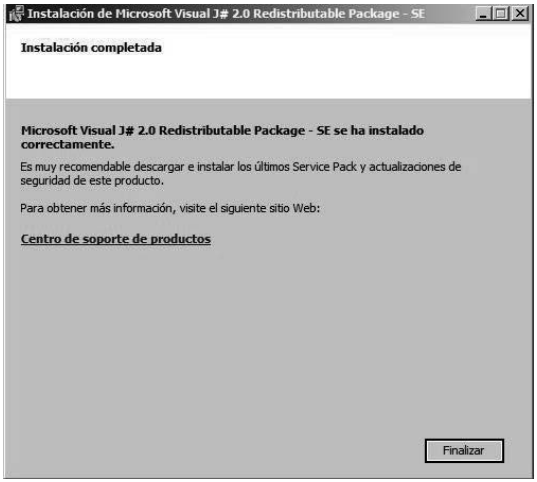

Figura No 111. Instalando Visual J# Parte 4.

Así termina nuestra instalación; ahora sí, instalamos Virtual Breadboard, para ello, ejecutamos Setup, y realizamos los siguientes pasos:

Figura No 112. Ícono de configuración.

Figura No 113. Setup de instalación de VirtualBreadBoard Parte 1.

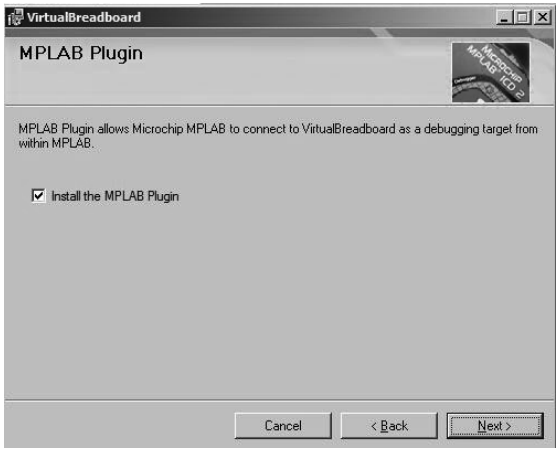

Figura No 114. Setup de instalación de VirtualBreadBoard Parte 2.

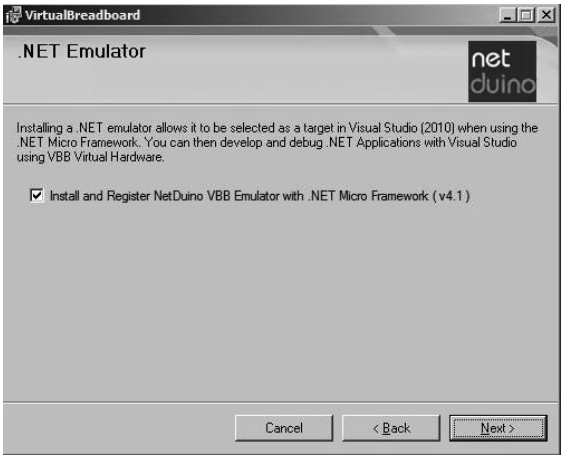

Figura No 115. Setup de instalación de VirtualBreadBoard Parte 3.

Elegimos la ubicación y quién lo puede usar: sólo yo (Just me) o todos (Everyone).

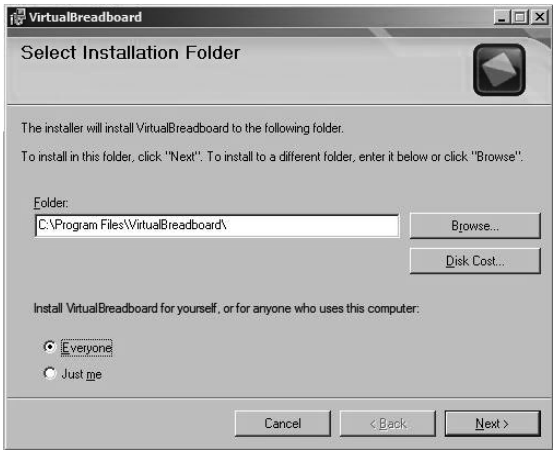

Figura No 116. Setup de instalación de VirtualBreadBoard Parte 4.

Confirmamos la instalación:

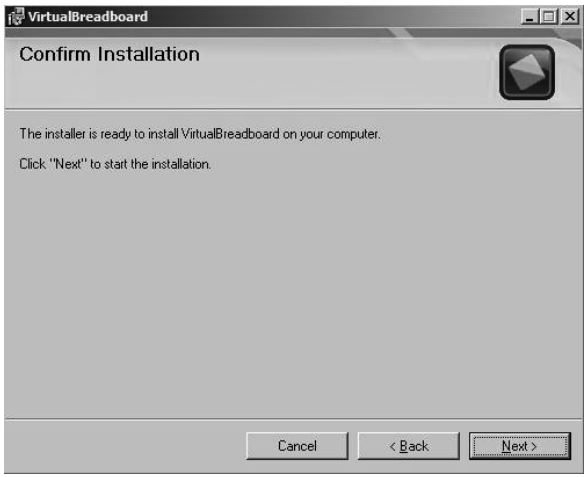

Figura No 117. Setup de instalación de VirtualBreadBoard Parte 5.

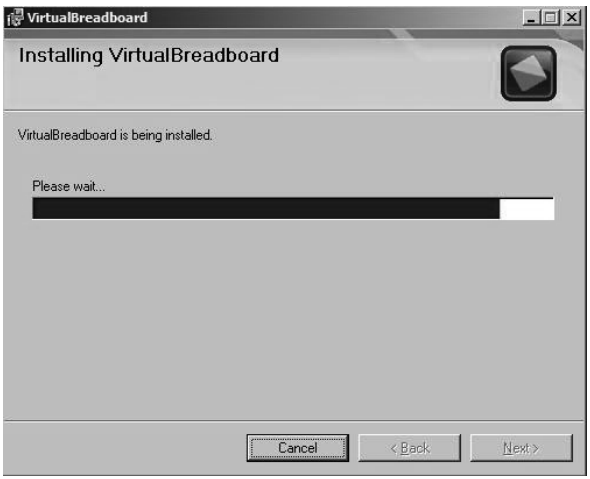

Figura No 118. Setup de instalación de VirtualBreadBoard Parte 6.

Y así queda la instalación completada:

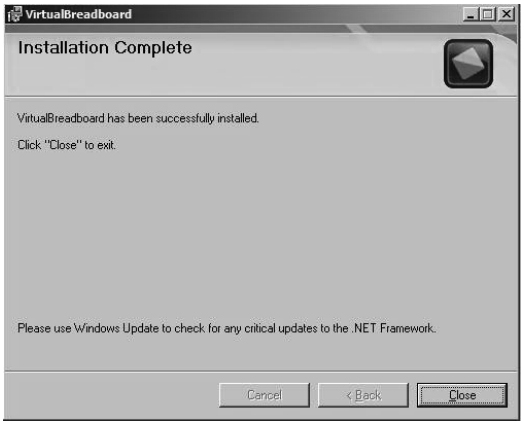

Figura No 119. Setup de instalación de VirtualBreadBoard Parte 7.

Para su funcionamiento, en el menú de inicio nos aparecerá el acceso rápido a la aplicación:

Figura No 120. Funcionamiento de VirtualBreadBoard Parte 1.

Y nos abrirá esta interface:

Figura No 121. Funcionamiento de VirtualBreadBoard Parte 2.

Ahora si usemos Virtual Breadboard: para ver simulaciones ya creadas, hacemos lo siguiente, como estamos trabajando la placa Arduino, ejecutamos Arduino; de nuevo Arduino y escogemos digital, y de la serie de opciones seleccionaremos Blink, eso nos llevará a esta interface donde se encuentra un simulador de la placa, de los LED, la consola de programación y errores, y además, todos los componentes que podemos utilizar.

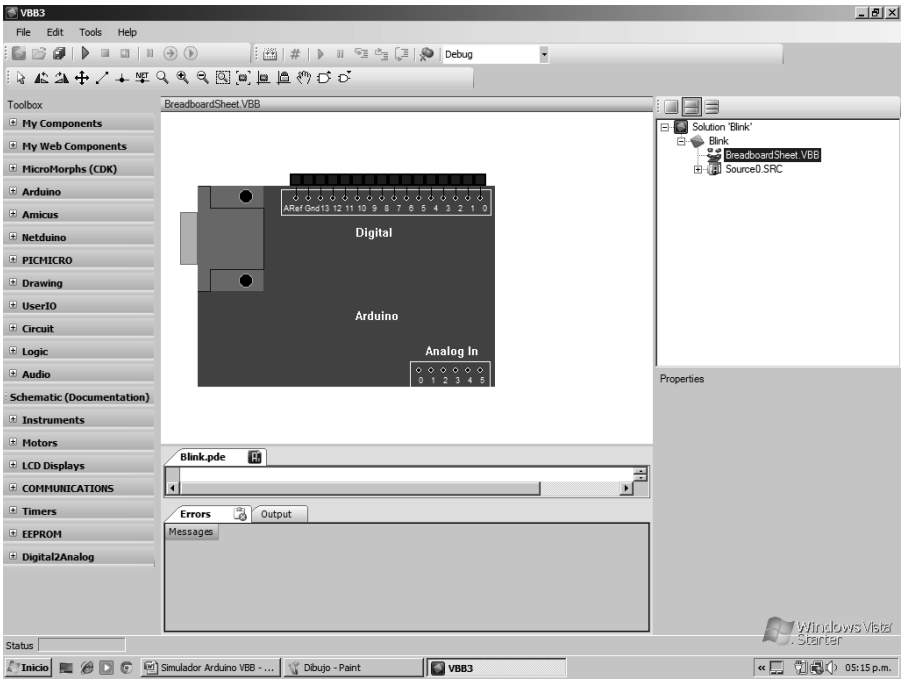

Figura No 122. Funcionamiento de VirtualBreadBoard Parte 3.

Al correr el programa,

Figura No 123. Funcionamiento de VirtualBreadBoard Parte 4.

Pulsando nuevamente el botón de Play, veremos el programa ejecutarse sobre la placa.

Figura No 124. Funcionamiento de VirtualBreadBoard Parte 5.

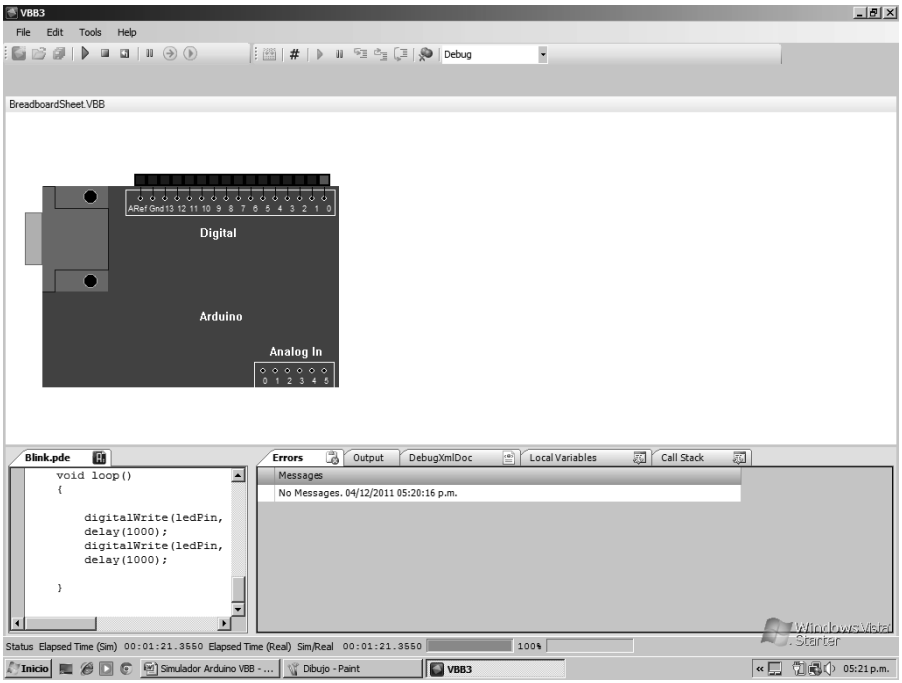

Figura No 125. Funcionamiento de VirtualBreadBoard Parte 6.

Se ve como el LED prende y apaga, como si estuviera sobre una placa real.

Luego, le damos stop,

Figura No 126. Funcionamiento de VirtualBreadBoard Parte 7.

y nos regresará a la ventana inicial.

Ahora entramos a File, y seleccionamos Close, solución para empezar de nuevo; luego, seleccionamos File nuevamente y escogemos New, y en la ventana de opciones escogemos New Project para empezar a trabajar nuestro propio proyecto.

Guardamos el proyecto, le damos el nombre que creamos conveniente, por ejemplo, este proyecto se llamara "Bandera".

En el toolbox, encontraremos la opción Arduino, escogeremos ArduinoStandar y arrastraremos la placa hasta nuestro simulador. En el mismo toolbox entramos a UserIO y arrastramos tres LED, que serán los tres colores de nuestra bandera colombiana.

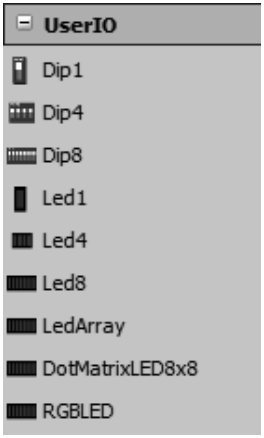

Figura No 127. Funcionamiento de VirtualBreadBoard Parte 8.

Después de poner nuestros tres LED, le cambiamos el color a cada uno en la barra de propiedades.

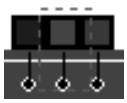

Figura No 128. Funcionamiento de VirtualBreadBoard Parte 9.

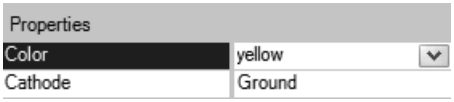

Figura No 129. Funcionamiento de VirtualBreadBoard Parte 10.

Ahora, procedemos a programar la placa:

```
void setup(){

    pinMode(10, OUTPUT);
    pinMode(9, OUTPUT);
    pinMode(8, OUTPUT);
}
void loop() {

    digitalWrite(10, HIGH);
    delay(500);
    digitalWrite(9, HIGH);
    delay(500);
    digitalWrite(8, HIGH);

    delay(2000);

    digitalWrite(10, LOW);
    digitalWrite(9, LOW);
    digitalWrite(8, LOW);

    delay(1000);
}
```

Figura No 130. Funcionamiento de VirtualBreadBoard Parte 11.

Y prendemos y apagamos nuestros LED de manera paulatina, dependiendo de un retardo.

Led

Figura No 131[88]. LED.

Partes del LED

Está compuesto por dos terminales, una terminal positiva (ánodo) y una terminal negativa (cátodo). La intensidad de su brillo depende de qué tanta corriente atraviese a través de él. Por lo general, los diodos tienen una caída de voltaje constante entre sus terminales cuando están polarizadas directamente, es decir, no importa qué tanta corriente atraviese l (luego de que no supere sus valores máximos), el voltaje entre las terminales será de 1.2V, por ejemplo.

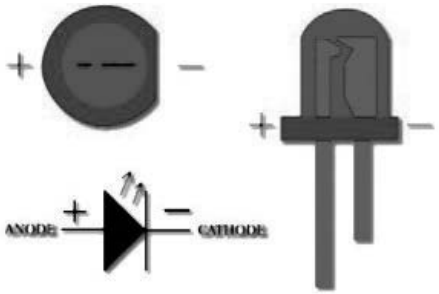

Figura No 132[89]. LED.

[88] LED. Disponible en: http://andreatecnologiacuarto.blogspot.com/

[89] LED. Disponible en: http://electroniccircuitsdiagram.com/led-power-supply

Funcionamiento

LED, Light Emisor Diode o Diodo Emisor de Luz, es un componente electrónico que emite un haz de luz cuando sus terminales están directamente polarizadas, de lo contrario no emitirá ningún haz de luz, ya que como su mismo nombre lo dice (efecto diodo), no permite el paso de la corriente.

Figura No 133[90]. LED.

LED multicolor

Algunos LED se componen de más de dos terminales, en la mayoría de los casos es porque el dispositivo es capaz de emitir más de un color de LED. Dependiendo de la terminal que se esté polarizando, encenderá el color deseado.

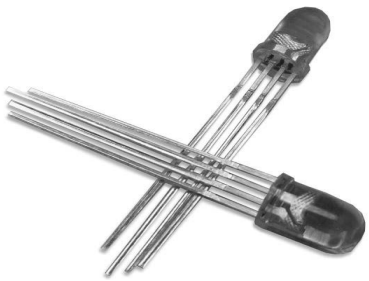

Figura No 134[91]. LED Multicolor.

[90] LED. Disponible en: http://www.custobots.com/products/standard-red-led-sled

[91] LED Multicolor. Disponible en: http://jj-iluled.es/cgi-bin/weblog_basic/index.php?cat=5

LED difuso

Es el tipo de LED comúnmente usado en el mercado, se caracteriza por tener un brillo bajo y opaco y de un solo color. Se recomienda su uso para señalización de eventos en un hardware.

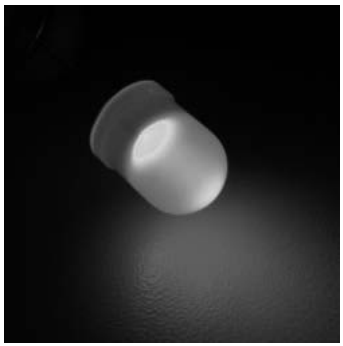

Figura No 135[92]. LED difuso.

LED bicolor

LED de dos terminales que enciende con el paso de la corriente en ambos sentidos. En un sentido de la corriente enciende de un color, y en el otro sentido, enciende de otro color.

Figura No 136[93]. LED bicolor

[92] LED Difuso. Disponible en: http://lista.mercadolivre.com.br/5x-led-difuso-laranja-lente

[93] LED BiColor. Disponible http://www.electronicamagnabit.com/tienda/35-led-5mm

LED infrarrojo

Diodos que emiten un haz de luz que se encuentra en el espectro de luz infrarroja imperceptible al ojo humano. Es usado para la transmisión de señales y comunicación inalámbrica. Un ejemplo claro de esto son los controles remotos. Este tipo de componente siempre debe trabajar de la mano con un fotosensor que sea capaz de detectar la emisión o ausencia del haz de luz infrarrojo.

Figura No 137[94]. LED infrarrojo.

LED de alta luminosidad

Se caracterizan por el alto brillo con el que pueden proyectar su haz de luz. Se recomienda su uso para iluminación, mas no para señalización de información en un hardware, ya que se puede apreciar su haz de luz al máximo es sobre superficies.

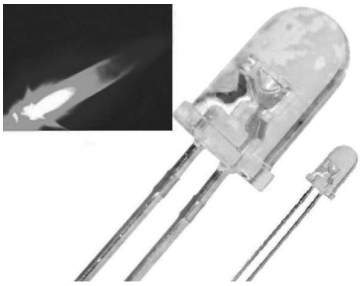

Figura No 138[95]. LED de alta luminosidad.

[94] LED Infrarrojo. Disponible en: http://www.gratisjuegos.org/descargar/usar-el-wiimote-joystick-de-wii-en-el-pc/

[95] LED de Alta Luminosidad. Disponible en: Imagen tomada de http://www.shoptronica.com/809-led-megabrillo-5mm-de-muy-alta-luminosidad.html

Manejo de Memoria

EEPROM - FLASH – SRAM

Figura No 139. Manejando la memoria

La unidad de manejo de memoria (MMU) es un dispositivo de hardware formado por un grupo de circuitos integrados responsable del manejo de los accesos a la memoria por parte de la unidad de procesamiento central -CPU.

Entre las funciones de este dispositivo se encuentran la traducción de las direcciones lógicas (o virtuales) a direcciones físicas (o reales), la protección de la memoria, el control de caché y, en arquitecturas de computadoras más simples (especialmente en sistemas de 8 bits), Bank switching.

Cuando la CPU intenta acceder a una dirección de memoria lógica, la MMU realiza una búsqueda en una memoria caché especial llamada Buffer de Traducción Adelantada (TLB, Translation Lookaside Buffer), que mantiene la parte de la tabla de páginas usadas más recientemente; esto es, se mantienen entradas de la tabla de páginas (llamadas PTE por sus siglas en inglés, Page Table Entry), donde se pueden rescatar las direcciones físicas correspondientes a algunas direcciones lógicas de forma directa.

Cuando la dirección requerida por la CPU se encuentra en el TLB, su traducción a dirección real o física es entregada, en lo que se conoce como 'acierto en el TLB' ('TLB hit'). En otro caso, cuando la dirección buscada no se encuentra en el TLB (fallo en el TLB), el procesador busca en la tabla de páginas del proceso utilizando el número de página como entrada a la misma.

En la entrada de la tabla de páginas del proceso se encuentra un bit de presencia que indica si la página buscada está en la memoria principal. Si el bit de presencia está activado, se carga esta PTE en el TLB y se devuelve la dirección física. En caso contrario, se informa al sistema operativo de la situación, mediante un fallo de página. Es el sistema operativo el encargado de realizar los ajustes necesarios (esto es, cargar la página en memoria física) usando uno de los algoritmos de reemplazo de páginas para continuar con la ejecución desde la instrucción que causó el fallo.

Un beneficio fundamental de la MMU es la posibilidad de implementar protección de memoria, evitando que los programas accedan a porciones de memoria prohibidas; se puede, por ejemplo, evitar que un programa acceda o modifique sectores de memoria de otros programas.

Memoria estática

La forma más fácil de almacenar el contenido de una variable en memoria durante todo el tiempo de ejecución es en la memoria estática o permanente. Sin embargo, no todas las variables pueden ser almacenadas estáticamente. Para que un objeto pueda ser almacenado en memoria estática, su tamaño (número de bytes necesarios para su almacenamiento) ha de ser conocido en términos de tiempo de compilación, como consecuencia de esta condición no podrán almacenarse en memoria estática:

1. Los objetos correspondientes a procedimientos o funciones recursivas, ya que en tiempo de compilación no se sabe el número de variables que serán necesarias.

2. Las estructuras dinámicas de datos tales como listas, árboles, etc., ya que el número de elementos que las forman no es conocido hasta que el programa se ejecuta.

Las técnicas de asignación de memoria estática son sencillas. A partir de una posición señalada por un puntero de referencia, se aloja la variable X y se avanza el puntero tantos bytes como sean necesarios para almacenar la variable X.

La asignación de memoria puede hacerse en tiempo de compilación y las variables estarán vigentes desde que comienza la ejecución del programa hasta que termina.

Memoria EEPROM

Figura No 140[96]. Memoria EEPROM.

EEPROM son las siglas de Electrically-Erasable Programmable Read-Only Memory (Memoria de solo lectura programable y borrable eléctricamente). Como su nombre lo indica, es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente, es decir, es una memoria no volátil.

[96] Imagen de lizykaryhermosas.blogspot.com

Las memorias de tipo EEPROM tienen como principal cualidad, permitir el almacenamiento y la sobre-escritura de datos por medio de los voltajes de operación norma de los circuitos electrónicos, además sostienen la información por muchos años sin fuente de alimentación.

Podemos encontrar circuitos integrados de memorias EEPROM paralelos, compatibles pin a pin con circuitos de memoria RAM o de memoria EPROM. Este tipo de memorias, precisamente por ser de interfaz paralela, tienen muchos pines externos por medio de los cuales recibe y entrega los datos y permite el direccionamiento de las distintas posiciones de almacenamiento. Debido a esto, los circuitos integrados son de gran tamaño físico, impidiendo ser utilizados en aplicaciones que requieran tamaño reducido.

Con las memorias EEPROM de interfaz serial, el control se ha reducido solamente a unos cuantos pines que son utilizados para entrada o salida de datos en forma serial (1 ó 2 pines), habilitación (1 pin), reloj de sincronismo (1 pin), direccionamiento de dispositivo (3 pines) que no existen en la interfaz paralela, y por último, los pines de alimentación del circuito (2 pines).

Los datos y la dirección de las posiciones de memoria utilizarán únicamente uno o dos pines, dependiendo del tipo de comunicación utilizada (dos o tres hilos). La velocidad de transferencia de datos puede variar desde los 100 KHz hasta los 600 MHz, dependiendo del tipo de memoria y del sistema de comunicación utilizados.

Características principales de la EEPROM

1. Se pueden conectar fácilmente con microprocesadores o microcontroladores, algunas de estas memorias tienen pines para realizar esta labor.

2. Transferencia de datos de manera serial, lo que permite ahorro del micro para dedicarlo a otras funciones.

3. El consumo de corriente es mucho menor que en las memorias que trabajan en paralelo.

Flash

Figura No 141[97]. Memoria flash.

La memoria flash es una tecnología de almacenamiento, derivada de la memoria EEPROM que permite la lecto-escritura de múltiples posiciones de memoria en la misma operación. Gracias a ello, la tecnología flash, siempre mediante impulsos eléctricos, permite velocidades de funcionamiento muy superiores frente a la tecnología EEPROM, primigenia que sólo permitía actuar sobre una única celda de memoria en cada operación de programación.

Este tipo de memoria suele ser usada en cámaras digitales, celulares, reproductores y PDA.

[97] Flash. Disponible en: Imagen de jesustecman.blogspot.com

SRAM

Figura No 142[98]. Memoria SRAM.

SRAM son las siglas de Static Random Aleatory Memory, que traducido es memoria estática de acceso aleatorio. La SRAM es un tipo de memoria basada en semiconductores que a diferencia de la memoria DRAM, es capaz de mantener los datos mientras esté alimentada, sin necesidad de un circuito de actualización. Sin embargo, sí son memorias volátiles, es decir que pierden la información si se les interrumpe la alimentación eléctrica.

Como ya se mecionó, estas memorias son de acceso aleatorio, lo que significa que las posiciones en la memoria pueden ser escritas o leídas en cualquier orden, independientemente de cual fuera la última posición de memoria accedida.

[98] Imagen de tweaktown.com

Aprendiendo lenguaje de programación Arduino

Parte I: Estructura

El lenguaje utilizado en la programación de la placa Arduino es llamado Processing, este es un lenguaje de programación y entorno de desarrollo integrado de código abierto basado en Java, de fácil empleo.

Estructura base. La estructura básica del lenguaje de programación de Arduino se compone de al menos dos partes. Estas dos partes necesarias, o funciones, encierran bloques que contienen declaraciones, estamentos o instrucciones.

```
void setup(){
  Instrucciones;}
void loop(){
  Instrucciones;}
```

Código Fuente No. 5

Void[99] que en español significa vacío, es una palabra reservada que se utiliza sólo para declarar funciones e indica que se espera que no devuelva información a la función donde fue llamada.

Setup() es la parte encargada de recoger la configuración, se invoca una vez sólo cuando el programa empieza, debe ser incluido en un programa aunque no haya declaración que ejecutar. Se utiliza para inicializar y/o configurar los pin, modo de trabajo de entradas y salidas o puerto serie. Ejemplo:

```
Void setup(){
PinMode (pin, OUTPUT); // Configura el pin como salida
       }
```

Código Fuente No. 6

Loop() es la función núcleo de cualquier programa, contiene el código que se ejecutará repetidamente; lee las entradas, activa las salidas, etc. Ejemplo:

```
void loop(){
digitalWrite(pin, HIGH); // Enciende el ´pin´
delay(3000); // espera tres segundos (3000 ms)
digitalWrite(pin, LOW); // Apaga el ´pin´
delay(4000); // espera cuatro segundos (4000 ms)}
```

Código Fuente No. 7

Sintaxis

Uso de llaves { }. Las llaves sirven para definir el principio y el final de un bloque de instrucciones. Se utilizan para las funciones de programación setup(), loop(), condicionales, etc. Una llave de apertura "{"siempre debe tener una llave de cierre "}", si no es así, el entorno de programación de Arduino incluye una herramienta de gran utilidad que arrojará los errores al compilar el programa.

Punto y coma ";". El punto y coma ";" es necesario para que el programa sepa cuándo termina cada instrucción y empieza otra. Olvidarse de un punto y coma es traducido como un error al momento de la compilación.

Bloque de comentarios /*… */. El bloque de comentarios es un área de texto que utilizamos para hacer referencia a la descripción de un código o del programa en general, aunque éste no es reconocido por el programa. Empiezan con "/*" y terminan con "*/" y puede abarcar un número indeterminado de líneas. Igualmente, si no se cierra el bloque de comentarios, se desencadenarán errores en el programa.

Línea de comentario //. Una línea de comentario se utiliza con frecuencia al final de una instrucción, ya sea para recordar algo o para proporcionar alguna información de la instrucción. Es precedida por "//" y termina con el inicio de la siguiente línea de código; y al igual que los bloques de código, son ignoradas por el programa.

Operadores aritméticos

Asignación "=". Guarda el valor de la derecha del símbolo igual en la variable que se encuentra en la parte izquierda, ejemplo:

Int variable = 0;

Suma (+) Resta (-) Multiplicación (*) División (/)

Los operadores aritméticos son suma, resta, multiplicación y división; éstos devuelven la suma, diferencia, producto o cociente de dos operandos (variables) teniendo en cuenta el tipo de dato con que se ha definido una variable, ejemplo:

Int var1 = 4; int var2 = 3; y int var3 = 0;

var3 = var1 / var2 donde el resultado (var3) será igual a 1 y no a 1.33, ya que las variables var1, var2 y var3 son de tipo entero y no se reconocen decimales con este tipo de datos.

Modulo " % ". Este operador se utiliza para saber el residuo de una división, ejemplo:

Int var1 = 7; int var2 = 7; y int var3 = 0;

Var3 = var1 % var2 donde el resultado (var3) será igual a 0.

Si var2 = 5, el resultado (var3) será igual a 2.

Operadores comparativos. Los operadores comparativos son usualmente una forma de comparar dos variables, se utilizan con frecuencia en los condicionales del tipo if para saber si una condición es cierta. Los operadores utilizados son:

Igual a == ejemplo: var1 == var2 // indicando que una variable es igual a otra.

Distinto de != ejemplo: var1 != var2 // indicando que la variable 1 es diferente a la variable 2.

Menor que < ejemplo: var1 < var2 // indicando que la variable 1 es menor que la variable 2.

Mayor que > ejemplo: var1 > var2 // indicando que la variable 1 es mayor que la variable 2.

Menor o igual que <= ejemplo: var1 <= var2 // indicando que la variable 1 puede ser menor o igual que la variable 2.

Mayor o igual que >= ejemplo: var1 >= var2 // indicando que la variable 1 puede ser mayor o igual que la variable 2.

Operadores booleanos o lógicos. Estos operadores se utilizan para comparar dos expresiones y, al igual que los operadores comparativos, se utilizan con frecuencia en los condicionales del tipo if, los operadores son:

&& (y), (AND) ejemplo: if (var1==0 && Var2>0) // indicando que la instrucción contenida en if sólo se ejecuta si se cumplen las dos condiciones de los operadores comparativos.

|| (o), (OR) ejemplo: if (var1==0 || Var2>0) // indicando que la instrucción contenida en if se ejecuta si se cumplen por lo menos una de las dos condiciones.

¡ (negación), (NOT) ejemplo: if (¡var1) // indicando que es verdadero si el valor de var1 es 0.

Operadores de composición. Los operadores de composición realizan operaciones matemáticas sin definir dos veces una misma variable, los operadores son:

++ Incremento ejemplo: var1++ // incrementa en uno el valor de la variable.

-- Disminución ejemplo: var1-- // disminuye en uno el valor de la variable.

+= Composición suma ejemplo: var1 += var2 // equivale a var1 = var1 + var2

-= Composición resta ejemplo: var1 -= var2 // equivale a var1 = var1 - var2

***= Composición multiplicación** ejemplo: var1 *= var2 // equivale a var1 = var1 * var2

/= Composición división ejemplo: var1 /= var2 // equivale a var1 = var1 / var2

Estructuras de control. Las estructuras de control[99] sirven para controlar la ejecución de instrucciones en el programa. Todas las estructuras de control tienen un único punto de entrada y un único punto de salida. Las estructuras que se utilizan en Arduino son:

If (Comparador Si). Es una sentencia condicional que se utiliza para saber si una condición se cumple, se emplea a través de los operadores comparativos, ejemplo:

```
If (var1 == 3) // Si la var1 es igual a 3 entonces  haga.
{
Haga
}
```

Código Fuente No. 8

If…else (Comparador si… sino). Esta sentencia nos da la opción de que si un condicional no se cumple se ejecutara otra instrucción, ejemplo:

```
If (var1 == 3) // Si la var1 es igual a 3 entonces  haga.
{
Haga A
}  else //sino
{
Haga B
}
```

Código Fuente No. 9

[99] Tomado de http://es.wikipedia.org/wiki/Estructuras_de_control

For (Contador). Es una declaración usada para repetir sentencias un determinado número de veces. Comprende tres partes encerradas en paréntesis: inicialización de la variable (ésta debe ser entera), condición, incremento o disminución separados por punto y coma (;) y encierra las sentencias entre llaves { }, ejemplo:

```
For (int i = 0; i< 10; i++) {
Haga
}
```

Código Fuente No. 10

Es importante saber que 0 (cero) es contado como una posición, eso quiere decir que el ciclo se repetirá 10 veces.

Switch case (Comparador múltiple). Switch es una sentencia encargada de encontrar el valor de una variable en el valor de un case (caso) y así ejecutar instrucciones. Ejemplo:

```
switch (var1) // donde var1 es una variable de tipo entero {
    case 1:
      //haga ,cuando var1 sea igual a 1
      break;
    case 2:
      //haga, cuando var1 sea igual a 2
      break;
    default:
      // si nada coincide, ejecuta el "default"
      // el "default" es opcional
  }
```

Código Fuente No. 11

While (Mientras que). Es un condicional que se ejecutará repetidamente mientras que la sentencia sea verdadera, ejemplo:

```
Var1 = 30
While (var1 > 10) {
//Haga
Var1 --;
}
```

Código Fuente No. 12

Do…while (Condicional hacer mientras que). Este condicional trabaja igual que el condicional while, con la diferencia de que la comparación se realiza al final, por lo tanto, esta sentencia se ejecutará por lo menos una vez, ejemplo:

```
Var1 = 30
Do {
       // haga
       Var1 --;
       } While (var1 > 10);
```

Código Fuente No. 13

Break (Salida). El termino break es utilizado para salir de una estructura de control sin importar la sentencia inicial.

Continue (Continuación). Dentro de un condicional iterativo, la sentencia continue omite el resto de sentencias y continúa con el resto de las iteraciones.

Return (Devolver). Return devuelve un valor a la función que lo llama, ejemplo:

```
Int devvalor () // método que recoge el valor
{
       // haga
       Return valor;
}
```

Código Fuente No. 14

Parte II: Funciones

Las funciones son bloques de códigos y sentencias que sólo se ejecutan cuando la función es llamada. Las funciones son creadas por un usuario con el fin de reducir el tamaño de un programa, así también, el de evitar realizar tareas repetitivas.

Las funciones cuando son invocadas necesitan parámetros de entrada que se encuentran dentro de paréntesis; ejemplo: Funcion(Parametro1,..,etc). Cabe agregar que pueden haber funciones que no necesiten parámetros como setup() o loop().

Las funciones incluidas en el IDE de Arduino que podemos utilizar son las siguientes:

Funciones de entradas/salidas digitales[100]

pinMode ( ). Configura el pin indicado para que se comporte como entrada o salida, esta función necesita de dos parámetros: el pin y el modo -que puede ser INPUT (entrada) o OUTPUT (salida)-, ejemplo:

```
pinMode (pin1, OUTPUT); // El pin 1 es una salida.
```

Código Fuente No. 15

digitalWrite ( ). Configura el voltaje del pin, HIGH (encendido) equivale a 5v o LOW (apagado) que equivale a 0v. Utiliza como parámetros el pin y el valor, ejemplo:

Si el pin 1 es configurado como salida entonces:

```
digitalWrite ( pin1, HIGH ); // El pin 1 se encenderá
```

Código Fuente No. 16

digitalRead ( ). Lee el valor digital de un pin especificado y lo devuelve a una variable, ejemplo:

[100] Tomado de http://Arduino.cc/es/Reference/

```
Var1 = digitalRead (1); // Lee el valor digital del pin 1
// Devolverá HIGH o LOW  y se lo asignara la variable var1
```

Código Fuente No. 17

Funciones de entradas/salidas analógicas[101]

analogReference ( ). Configura el voltaje de referencia usado por la entrada analógica. El tipo de referencia puede ser:

Default. Es el valor de referencia analógico que viene por defecto; de 5 voltios en placas Arduino de 5v y de 3.3 voltios en placas Arduino que funcionen con 3.3v.

Internal. Es una referencia de tensión interna de 1.1 voltios en el ATmega168 o ATmega328 y de 2.56 voltios en el ATmega8.

External. Se usará una tensión de referencia externa que tendrá que ser conectada al pin AREF.

analogRead ( ). Lee el valor de tensión de un pin definido como entrada analógica. Los pines analógicos a diferencia de los digitales su modo siempre es de entrada (INPUT). Ejemplo:

```
Valor = analogRead (pin); // Asigna a valor lo que lee.
```

Código Fuente No. 18

analogWrite ( ). Esta función lee un valor analógico PWM (Modulación por ancho de pulso; pulse-width modulation) a uno de los pin de Arduino marcados como PWM. Necesita dos parámetros, el pin en el cual se quiere generar la señal PWM y el ciclo de trabajo que oscila entre 0 y 255, ejemplo:

```
Valor = 255;
analogWrite (pin1, valor);
```

Código Fuente No. 19

[101] Tomado de http://Arduino.cc/es/Reference/; 3 de diciembre de 2011.

Funciones de Entradas/Salidas Avanzadas[102]

Tone ( ). Esta función genera una onda cuadrada de la frecuencia especificada en un pin; sus parámetros son: el pin en el que se genera el tono, la frecuencia del tono en hercios y la duración del tono en milisegundos.

```
tone (pin1, frecuencia, duración); // La duración es opcional.
```

Código Fuente No. 20

noTone ( ). Esta función detiene la generación de la señal cuadrada que se activa al hacer uso de la función tone(). Esta función no tiene efecto si no se está generando ningún tono.

```
noTone (pin1); // Se detiene la señal del pin1
```

Código Fuente No. 21

shiftOut ( )[103]. Desplaza un byte de datos bit a bit. Empieza desde el bit más significante (el que se encuentra más a la izquierda) o el menos significante (el que está más a la derecha). Cada bit se escribe siguiendo su turno en un pin de datos, después de conmutar un pin de reloj (señal de reloj) para indicar que el bit está disponible, sus parámetros son:

pinDatos. Es el pin del cual extraer cada bit (int).

pinReloj. Es el pin que hay que conmutar cada vez que a un pinDatos le ha sido enviado el valor correcto (int).

ordenBits. Establece en qué orden desplazar los bits; si hacia el MSBFIRST (bit más significante primero) o hacia el LSBFIRST (bit menos significante primero).

Valor. Son los datos que rotar (byte). Ejemplo:

[102] Tomado de http://Arduino.cc/es/Reference/ 3 de diciembre de 2011

[103] Tomado de http://arduino.cc/es/Reference/ShiftOut

```
shiftOut(pinDatos, pinReloj, ordenBits, valor);
```

Código Fuente No. 22

El *pinDatos* y *pinReloj* deben estar ya configurados como salida con una llamada previa a pinMode().

pulseIn ( ). Esta función lee el pulso en un pin, ya sea alto (HIGH) o bajo (LOW). Utiliza como parámetros el pin del cual se lee la pulsación, el tipo de pulso y el tiempo en microsegundos es opcional, ejemplo:

```
pulseIn (pin, tipo de pulso, tiempo);
```

Código Fuente No. 23

Tiempo:[104]

Millis ( ). Esta función devuelve el número de milisegundos transcurridos desde el inicio del programa en Arduino o la última vez que se pulsó el botón reset hasta el momento actual; esta función no necesita parámetros.

```
Valor = millis ( );
```

Código Fuente No. 24

Micros ( ). Esta función devuelve el número de microsegundos transcurridos desde el inicio del programa en Arduino. Esta función no necesita parámetros.

```
Valor = micros ();
```

Código Fuente No. 25

[104] Tomado de Arduino.cc/es/Reference/

Delay ( ). Esta función detiene el programa la cantidad de milisegundos que se le indique en el parámetro, ejemplo:

```
Delay (3000); // Espera 3 segundos.
```

Código Fuente No. 26

delayMicroseconds ( ). Esta función detiene el programa la cantidad de microsegundos que se le indique en el parámetro, ejemplo:

```
.  DelayMicroseconds (300); //Espera 300 microsegundos.
```

Código Fuente No. 27

Matemáticas[105]

min( ). Calcula el mínimo de dos números de cualquier tipo de dato; sus parámetros son dos números, ejemplo:

```
Valor = min (dato1, dato2); // Si  dato1 es menor que dato2 a
valor se le asignara dato1, y si dato1 es mayor que dato2 a valor
se le asignara dato2.
```

Código Fuente No. 28

max(). Calcula el máximo de dos números de cualquier tipo de dato; sus parámetros son dos números, ejemplo:

```
Valor = max (dato1, dato2); // Si  dato1 es menor que dato2 a
valor se le asignara dato2, y si dato1 es mayor que dato2 a valor
se le asignara dato1.
```

Código Fuente No. 29

abs( ). Esta función calcula el valor absoluto de un número, devuelve el número si éste es mayor que cero y devuelve el número negativo si éste es menor que cero.

[105] Tomado de Arduino.cc/es/Reference/

```
Valor = abs (num);
```

Código Fuente No. 30

constrain( ). Esta función restringe un n**ú**mero a un rango definido; sus parámetros son tres n**ú**meros: el n**ú**mero a restringir y los dos números del rango, ejemplo:

```
Valor = Constrain (num, 20, 40); // Se limita a  número entre 20
y 40, si numero el numero esta en el rango, es asignado a valor,
si el numero es mayor que el rango a valor se le asigna 40 y si
el número es menor que el rango a valor se le asignará 20.
```

Código Fuente No. 31

Map ( )[106]. Esta función re-mapea un número desde un rango hacia otro. Esto significa que un valor (value) con respecto al rango fromLow-fromHigh **será mapeado al rango** toLow-toHigh, donde:

value: el número (valor) a mapear.

fromLow: el límite inferior del rango actual del valor.

fromHigh: el límite superior del rango actual del valor.

toLow: límite inferior del rango deseado.

toHigh: límite superior del rango deseado.

```
val = map(val, 0, 1023, 0, 255);
```

Código Fuente No. 32

pow( ). Esta función eleva un número al exponente deseado; sus parámetros son dos: base y exponente, ejemplo:

[106] Tomado de http://arduino.cc/es/Reference/Map

```
valor = pow (2,3) // A valor se le asignara el resultado de 2
elevado a la 3, por lo tanto valor será 8.
```

Código Fuente No. 33

sq( ). Esta función eleva un número al cuadrado, ejemplo:

```
Valor = sq (2); // Valor será 4.
```

Código Fuente No. 34

sqrt(). Esta función determina la raíz cuadrada de un número, ejemplo:

```
Valor = sqrt (25); // Valor será igual a 5.
```

Código Fuente No. 35

Trigonometría

Sin ( ). Calcula el seno de un ángulo en radianes, ejemplo:

```
Valor = sin (90); // A valor le será asignado 1
```

Código Fuente No. 36

Cos ( ). Calcula el coseno de un ángulo en radianes, ejemplo:

```
Valor = cos (180); // A valor le será asignado -1.
```

Código Fuente No. 37

Tan ( ). Calcula la tangente de un ángulo en radianes, ejemplo:

```
Valor = tan (45); // A valor le será asignado 1.
```

Código Fuente No. 38

Parte III:
Diccionario de palabras reservadas del IDE de Arduino.

Las palabras reservadas del IDE de Arduino son las constantes, variables y funciones que se definen en el lenguaje de programación de Arduino. No se deben usar estas palabras clave para nombres de variables, algunas de estas palabras tienen la propiedad de cambiar de color cuando las escribimos en el IDE.

Constantes[107]

Tabla No 3.
Constantes.

HIGH	Nivel alto
LOW	Nivel bajo
INPUT	Entrada
OUTPUT	Salida
SERIAL	Puerto serie, comunicación entre la placa Arduino y el ordenador.
DISPLAY	Visualización
PI	Pi equivale a 3,14
HALF_PI	Equivale a medio Pi, 1,570796…
TWO_PI	Equivale a dos Pi, 6.28318…
QUARTER_PI	Equivale a un cuarto de Pi, 0.7853982
LSBFIRST	Bit menos significante primero
MSBFIRST	Bit más significante primero
CHANGE	Para disparar la interrupción en cualquier momento que el pin cambie de valor.
FALLING	Para cuando el pin cambie de valor alto a bajo.
RISING	Para disparar la interrupción cuando el pin cambie de valor alto a bajo.
False	Falso
True	Verdadero
Null	Nulo

[107] Tomado de http://arduino.cc/es/Reference 2 de diciembre de 2011

Variables de designación de puertos y constantes[108]

Tabla No 4.
Variables de designación de puertos constantes.

DDRB	Dirección del registro de datos del puerto B
PINB	Registro del pin de entrada del puerto B
PORTB	Registro de datos del puerto B
PB0	
PB1	
PB2	
PB3	
PB4	
PB5	
PB6	
PB7	
DDRC	Dirección del registro de datos del puerto C
PINC	Registro del pin de entrada del puerto C
PORTC	Registro de datos del puerto C
PC0	
PC1	
PC2	
PC3	
PC4	
PC5	
PC6	
PC7	
DDRD	Dirección del registro de datos del puerto D
PIND	Registro del pin de entrada del puerto D
PORTD	Registro de datos del puerto D
PD0	
PD1	

[108] Tomado de http://arduino.cc/es/Reference 2 de diciembre de 2011

PD2	
PD3	
PD4	
PD5	
PD6	
PD7	
Private	Privado
Protected	Protegido
Public	Público
Return	Devolver
Short	Corto (Corto circuito)
Signed	Signo
Static	Estático
Switch	Interruptor
Throw	Arrojar
Try	Tratar de
Unsigned	Sin signo
Void	Declara funciones

Otras[109]

Tabla No 5.
Variables de designación de puertos constantes.

Abs	Absoluto (Función; calcula el valor absoluto de un número)
Acos	Arcocoseno
Asin	Arcoseno
Atan	Arcotangente
atan2	Arcotangente de dos parámetros

[109] Tomado de http://arduino.cc/es/Reference 2 de diciembre de 2011

case	Caso
Ceil	Menor entero no menor que el parámetro
Constrain	Restringir a
Cos	Coseno
Delay	Espera (tiempo)
loop	Función núcleo
max	Máximo
millis	Milisegundos
min	Mínimo
New	Nuevo
Setup	Recoge la configuración (Sólo se invoca al iniciar el programa)
Sin	Seno
Sq	Cuadrado
Sqrt	Raíz cuadrada
Tan	Tangente
This	Éste
While	Condicional mientras que
Begin	Pne vvelocidad a la comunicación
Read	Leer
Print	Imprime
Write	Escribe
Peek	Función que devuelve el primer byte de entrada de datos
Flush	Función que vacía el buffer de entrada de datos en serie
Println	Imprime
Available	Función que devuelve el número de bytes disponibles para ser leídos por el puerto serie
digitalWrite	Escritura del pin digital
digitalRead	Lectura del pin digital
PinMode	Modo del pin
analogRead	Lectura del pin análogo
analogWrite	Escritura del pin análogo
attachInterrupts	Interrupción externa

detachInterrupts	Apaga la interrupción
Tone	Tono de un pin
noTone	Detiene el tono del pin
pulseIn	Lee un pulso
shiftOut	Cambio
Map	Función que re-mapea un rango de valores a otro
Pow	Eleva un valor dado al exponente
Max	Máximo
Min	Mínimo
lowByte	Extrae un byte de derecha
highByte	Extrae un byte de izquierda
bitRead	Lee un bit
bitWrite	Escribe un bit
bitSet	Cambia un bit
bitClear	Borra un bit
Bit	Calcula el valor de un bit
randomSeed	Inicializa el generador de números
Random	Genera números
Class	Clase
Default	
Do	Hacer
Esp	
delayMicrosec-onds	
Floor	
log	

Tipos de datos[110]

Boolean. Sólo contiene valores que pueden ser TRUE o FALSE (verdadero o falso).

Byte. Número entero de ocho bits.

Char. Almacena un valor de carácter (letra) entre comillas simples.

Double. Número decimal.

Long. Número entero de tamaño extendido.

Float. Número decimal.

Int (integer). Número entero.

String. Tipo de caracteres extendido entre comillas.

Array. Matriz de variables accedidas mediante un número de índice.

Unsigned Int. Sólo almacena enteros positivos.

Unsigned Long. Sólo almacena enteros positivos de tamaño extendido.

Funciones de conversión

Char(). Convierte un valor de cualquier tipo a un tipo de dato char.

byte(). Convierte un valor de cualquier tipo a un tipo de dato byte.

int(). Convierte cualquier valor de tipo numérico a un tipo de dato entero.

long(). Convierte un valor de cualquier tipo a un tipo de dato long.

float(). Convierte un valor de cualquier tipo a un tipo de dato float.

[110] Tomado de http://arduino.cc/es/Reference 3 de diciembre de 2011

Potenciómetro

Figura No 143[111]. Potenciómetro.

¿Qué es el potenciómetro?

Es un elemento electrónico que cumple con las propiedades de un resistor y cuyo valor de resistencia es variable.

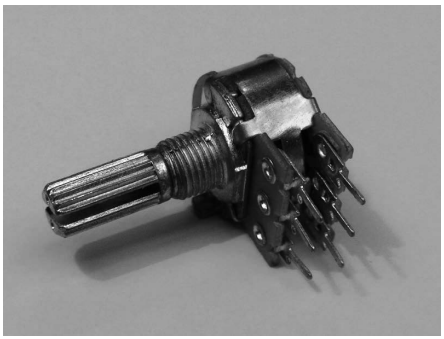

Figura No 144[112]. Potenciómetro 1.

[111] Potenciómetro. Disponible en: http://es.wikipedia.org/wiki/Archivo:Potentiometer.jpg

[112] Tipos de Potenciómetro. Disponible en: http://construyasuvideorockola.com/imagenes/curso/Electronica/potenciometro_doble.jpg

Partes del potenciómetro

Generalmente, los potenciómetros cuentan con 3 terminales y un eje variador de su resistividad.

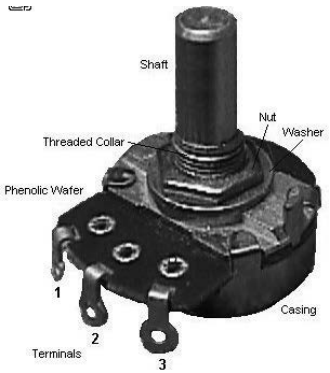

Figura No 145[113]. Partes del potenciómetro

Funcionamiento

Estos elementos al igual que los resistores tienen como función impedir el paso de la corriente; esto lo realiza de manera variable mediante las conexiones entre las terminales y el eje variador de la resistividad máxima en cierto porcentaje.

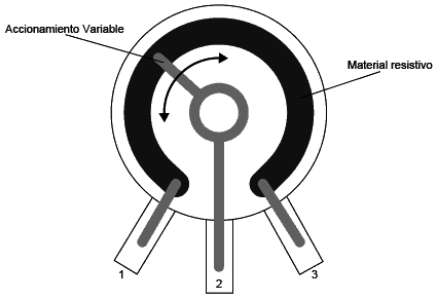

Figura No 146[114]. Funcionamiento del potenciómetro

[113] Partes del Potenciómetro. Disponible en: Imagen de http://asterion.almadark.com/wp-content/uploads/2009/12/pots-f5.jpg

[114] Funcionamiento del potenciómetro. Disponible en: http://www.factoriadoson.com/wp-content/gallery/potes/potenciometro-funcionamiento.jpg

Características

La característica que lo hace especial y fundamental en los circuitos, es que le permite variar la resistividad del elemento, y también, de manera indirecta, lo hace con la intensidad de corriente y la diferencia de potencial dentro del circuito.

Figura No 147[115]. Características del potenciómetro

Usos comunes

Los potenciómetros normalmente son usados en circuitos que manejan poca corriente y son utilizados como elementos de control de ciertas etapas dentro de los circuitos, como por ejemplo, en los circuitos de audio en el control de volúmen, o en los circuitos de contraste y/o iluminación de los televisores para variarles dichas características.

[115] Características del Potenciómetro. Disponible en: http://www.potenciometros.es/img/control_potentiometers.jpg

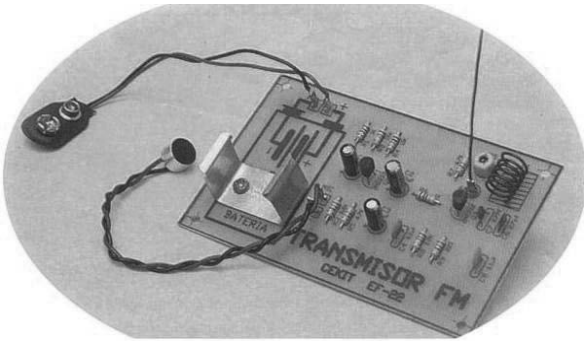

Figura No 148[116]. Usos del potenciómetro de la radio FM.

Tipos de potenciómetros

Potenciómetros rotatorios. Se controlan por medio del giro de su eje rotatorio. Son los más habituales pues son de larga duración y ocupan poco espacio.

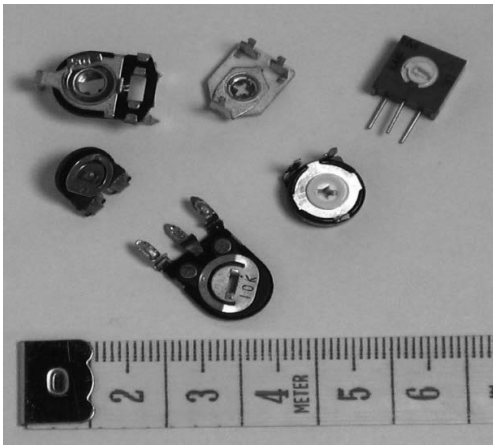

Figura No 149[117]. Potenciómetros rotatorios.

[116] Usos del potenciómetro de la Radio FM. Disponible en: http://www.infoartec.com.ar/ELECTR%C3%93NICA%20CIRCUITOS%20DIVERSOS_archivos/image002.jpg

[117] Potenciómetros Rotatorios. Disponibles en: http://upload.wikimedia.org/wikipedia/commons/8/87/PCB_variable_resistors.jpg

Potenciómetros deslizantes. Poseen una pista resistiva recta y un cursor que le permite variar su resistividad a lo largo de ésta. Son más frágiles que los rotatorios y ocupan más espacio. Además suelen ser más sensibles al polvo.

Figura No 150[118] Potenciómetros deslizantes.

Potenciómetros múltiples. Son varios potenciómetros situados sobre sus ejes coaxiales, de modo que ocupan menos espacio que los anteriores. Se usaban en autorradios e instrumentación, entre otros.

[118] Potenciómetros Deslizantes. Disponibles en: http://upload.wikimedia.org/wikipedia/commons/3/35/Faders.jpg

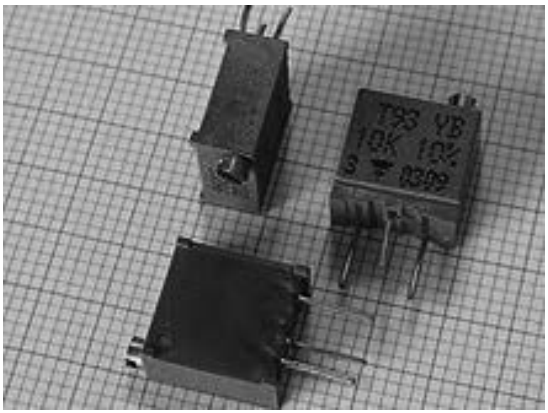

Figura No 151[119]. Potenciómetros múltiples.

Práctica con potenciómetro

Esta es una práctica sencilla donde se puede observar el funcionamiento de un potenciómetro. Aumentaremos y disminuiremos la intensidad de luz de un LED según aumentemos o disminuyamos el valor de la resistencia del potenciómetro. La fuente de voltaje la tomaremos de una de las salidas digitales del Arduino.

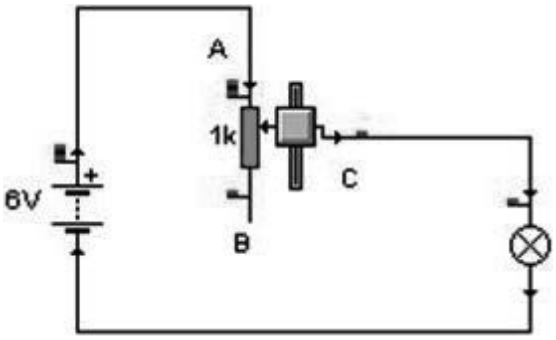

Figura No 152. Práctica con potenciómetro.

[119] Potenciómetros Múltiples. Disponibles en: http://upload.wikimedia.org/wikipedia/commons/thumb/0/07/Potenci%C3%B3metros_%22de_precisi%C3%B3n%22_sobre_papel_milimetrado.JPG/220px-Potenci%C3%B3metros_%22de_precisi%C3%B3n%22_sobre_papel_milimetrado.JPG

Prácticas con Arduino

Parte I

Observación: Algunas prácticas de esta cartilla han sido recogidas en la página oficial de Arduino: www.arduino.cc/

Práctica 1.1

Luces navideñas. Esta práctica consta de hacer parpadear los LED de los pines 2 al 7 en secuencia para crear el efecto de luces navideñas.

Esta práctica se ha escogido para aprender a programar secuencialmente, ya que sólo se trabajará con la función digitalWrite para cambiarle los estados a los pines y así lograr un efecto parpadeante, y la función delay como temporizador.

En el montaje se hace uso de 6 LED y 6 resistencias conectadas respectivamente una a una, además de tener puentes para hacer un sólo polo a tierra -GND.

Esquema:

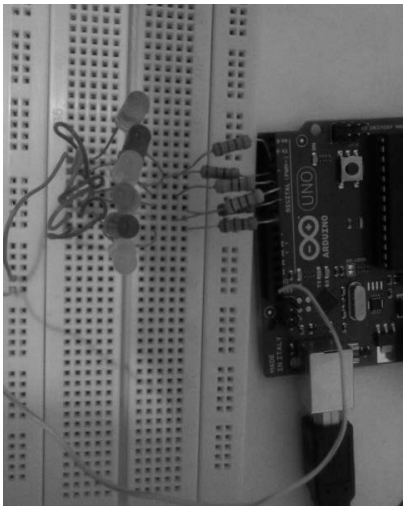

Figura No 153. Práctica Arduino 1.1

En nuestro IDE de Arduino tendremos:

```
int pin2 = 2;
int pin3 = 3;
int pin4 = 4;
int pin5 = 5;
int pin6 = 6;
int pin7 = 7;
int timer = 100; // Temporizador
void setup(){
pinMode(pin2, OUTPUT); // Configuración de los pines como salida
pinMode(pin3, OUTPUT);
pinMode(pin4, OUTPUT);
pinMode(pin5, OUTPUT);
pinMode(pin6, OUTPUT);
pinMode(pin7, OUTPUT);}
void loop() {
digitalWrite(pin2, HIGH); // Enciende y apaga secuencialmente
los LED
delay(timer);
digitalWrite(pin2, LOW);
delay(timer);
digitalWrite(pin3, HIGH);
delay(timer);
digitalWrite(pin3, LOW);
delay(timer);
digitalWrite(pin4, HIGH);
delay(timer);
digitalWrite(pin4, LOW);
delay(timer);
digitalWrite(pin5, HIGH);
```

```
delay(timer);
digitalWrite(pin3, LOW);
delay(timer);
}
```

Código Fuente No. 39

Práctica 1. 2

En esta práctica se hará lo mismo que en la anterior, sólo que la cantidad de código será reducida, porque se utilizará un vector, y el ciclo for.

En nuestro IDE de Arduino tendremos:

```
int pinArray[] = {2, 3, 4, 5, 6, 7}; // Define el array de pines
int count = 0; // Contador
int timer = 100; // Temporizador
void setup(){
for (count=0;count<6;count++){ // Configuramos todos los pines
pinMode(pinArray[count], OUTPUT);
} }
void loop() { // Enciende y apaga secuencialmente los LED
for (count=0;count<6;count++) { // utilizando la secuencia de
control
digitalWrite(pinArray[count], HIGH); // Recorrido de ida
delay(timer);
digitalWrite(pinArray[count], LOW);
delay(timer);}
```

```
for (count=5;count>=0;count--) {
digitalWrite(pinArray[count], HIGH); // Recorrido de vuelta
delay(timer);
digitalWrite(pinArray[count], LOW);
delay(timer);
} }
```

Código Fuente No. 40

Práctica 1. 3

En nuestra IDE de Arduino:

```
int pinArray[] = {2, 3, 4, 5, 6, 7}; // vector de pin´s
int count = 0; // Contador
int timer = 30; // Temporizador
void setup(){
for (count=0;count<6;count++) { // Configuramos todos los pin´s
juntos
pinMode(pinArray[count], OUTPUT);
} }
void loop() {
for (count=0;count<5;count++) { // Enciende los LED creando una
estela visual
digitalWrite(pinArray[count], HIGH);
delay(timer);
digitalWrite(pinArray[count + 1], HIGH);
delay(timer);
digitalWrite(pinArray[count], LOW);
delay(timer*2);}
```

Código Fuente No. 41

Parte II

Observación: Algunas prácticas de esta cartilla han sido recogidas en la página oficial de Arduino: www.arduino.cc/

Práctica 2. 1

Generador de notas musicales. Generar 8 notas musicales por el pin 11 de Arduino. Se debe crear un array (vector) de datos compuesto por los valores correspondientes a las 8 notas que se pretende sacar:

int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956};

Se deben definir también el tiempo de pausa entre nota y nota y el tiempo de pausa de fin de secuencia de notas:

int tnota=100; // Duracion de la nota

int pausa=100; // Tiempo de pausa entre una nota y otra

Las iteraciones para el recorrido de las 8 notas se realizan con un ciclo for:

for(n=0;n<8;n++)

El tiempo de activado y desactivado de la salida del parlante también se resuelve con

un ciclo for:

for(m=0;m<=tnota;m++){

Esquema:

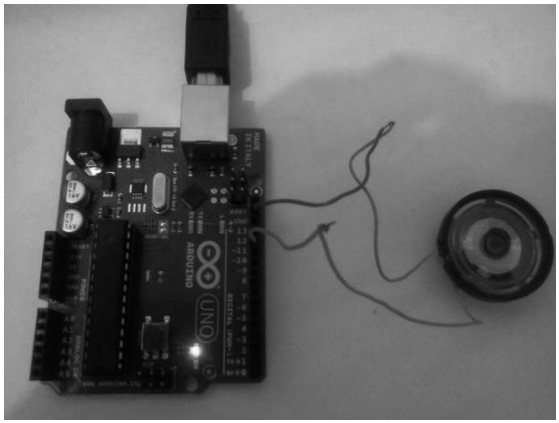

Figura No 154. Esquema Práctica Arduino 2.1

El parlante se conectará al pin 11 y a tierra GND.

En nuestro IDE de Arduino tendremos:

```
int parlante=11; // Se declara la variable parlante como el pin
11.
int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956}; //
vector con los tiempos que corresponden a las distintas notas
int n=0; // Variable  que se utiliza para el ciclo for del
recorrido de las notas.
int m= 0; // Variable que se utiliza para el ciclo for del tiempo
de activado y desactivado de la salida del parlante
int tnota=200; // Duración de la nota
int pausa=200; // Duración de la pausa
void setup() {
pinMode(parlante,OUTPUT);// pin 11 como modo de salida }
void loop(){
for(n=0;n<8;n++){ // Ciclo que recorre el vector con las duraciones
de los pulsos de cada nota
for(m=0;m<=tnota;m++){
digitalWrite(parlante,HIGH); // El parlante suena
```

Código Fuente No. 42

Práctica 2. 2

Timbre con pulsador. Se trata de realizar un timbre a través de un parlante, teniendo como salida el pin 11, que emita dos tonos recogidos de una colección de ocho tonos; por ejemplo, el tono 3 y el tono 7. El timbre se activa mediante un pulsador conectado en el pin 5 (entrada digital).

Utilizaremos las notas anteriores y escogeremos el tono 3 equivalente a 1432 y el tono 7 equivalente a 956.

Esquema:

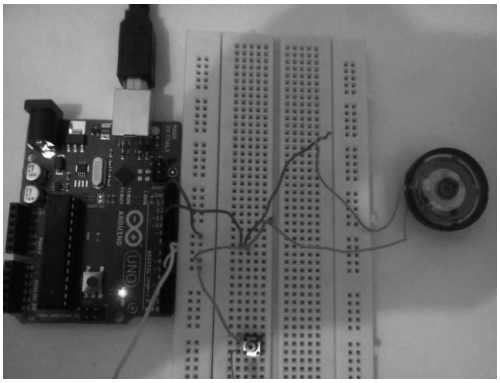

Figura No 155. Esquema Práctica Arduino 2. 2

En nuestro IDE de Arduino tendremos:

```
int notas[] = {1915, 1700, 1519, 1432, 1275, 1136, 1014, 956}; //
definición de vector de 8 notas
int puls=5; // designación del pulsador de llamada
int parlante=11; // designación de la salida hacia el parlante
int tnota=150;
int n=0;
int m=0;
void setup (){
for(n=0;n<4;n++){
pinMode(parlante,OUTPUT);
pinMode(puls,INPUT);
}
}
// Se crea la función nota
void nota() { // rutina que genera los tonos de llamada
for(m=0;m<=tnota;m++){
digitalWrite(parlante,HIGH);
delayMicroseconds(notas[n]);
digitalWrite(parlante,LOW);
delayMicroseconds(notas[n]);
}
```

```
}
void loop(){
if(digitalRead(puls)==HIGH){
n=3; //elegimos la primera nota del timbre
nota(); //que aquí es la tercera de la cadena
```

Código Fuente No. 43

Práctica 3

Encendido y apagado de un LED de manera analógica. Se trata de enviar hacia el pin 4 un valor analógico descendente y ascendente **cíclicamente comprendido entre 0 y 255, con incrementos de 5.**

para un pin analógico, su modo siempre es de salida -OUTPUT, por lo tanto, no es necesario que se le defina el modo.

Esquema:

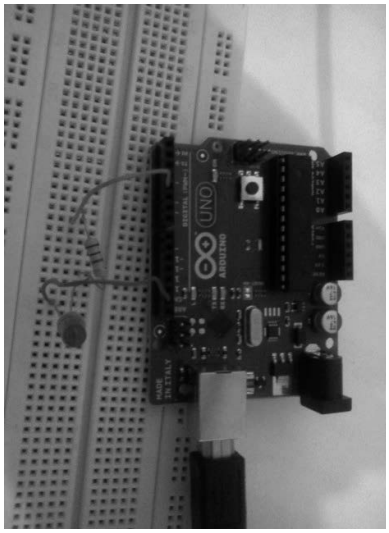

Figura No 156. Esquema Práctica Arduino 2. 3

En nuestro IDE de Arduino tendremos:

```
int value = 0; // Valor a sacar por la salida analógica del pin 4
int ledpin = 4; // Salida analógica pin 4
void setup()
{
  // Analógico siempre es salida, por eso no se define el modo del pin
}
void loop()
{
for(value = 0 ; value <= 255; value+=5) // Variación de la variable de salida entre el MIN y MAX
{
analogWrite(ledpin, value); // Enviar valor a la salida (entre 0 y 255)
delay(50); // Esperar 50 ms para ver el efecto de variación
}
for(value = 255; value >=0; value-=5) // Variación de la variable de salida entre MAX y MIN
{
analogWrite(ledpin, value);
delay(50);
}
}
```

Código Fuente No. 44

Parte III

Observación: Algunas prácticas de esta cartilla han sido recogidas en la página oficial de Arduino: www.arduino.cc/

Práctica 3. 1

Potenciómetro, lectura de señal analógica. Esta práctica consiste en encender y apagar un LED por medio de un potenciómetro conectado a la parte analógica de la placa Arduino. El tiempo que el LED parpadear**á** dependerá de la lectura análoga por parte del potenciómetro.

Esquema:

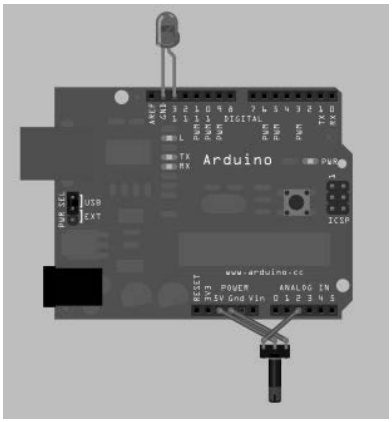

Figura No 157. Esquema general de placas de Arduino.

El LED se encontrará ubicado en el pin 13 de la señal digital, el potenciómetro ir**á** distribuido de la siguiente manera: la primera terminal irá conectada a tierra (GND), la segunda terminal ir**á** conectada a la entrada analógica 2 y la tercera terminal estará conectada a la salida de 5 voltios.

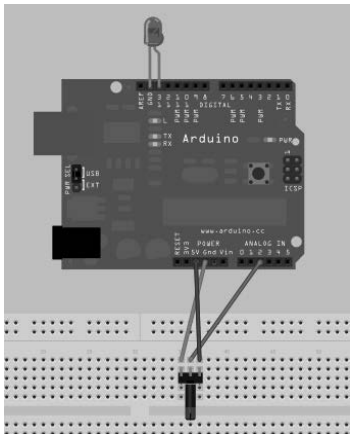

Figura No 158. Placa base.

En nuestro IDE de Arduino tendremos:

```
int potPin = 2; // seleccionar el pin de entrada analógico para el
potenciómetro
int ledPin = 13; // seleccionar el pin de salida digital para el LED
int val = 0; // variable para almacenar el valor capturado desde
el sensor
void setup() {
pinMode(ledPin, OUTPUT); // declara el ledPin en modo salida }
void loop() {
val = analogRead(potPin); // lee el valor del sensor
digitalWrite(ledPin, HIGH); // enciende LED
delay(val); // detiene el programa por un tiempo "val"
digitalWrite(ledPin, LOW); // apaga el LED
delay(val); // detiene el programa por un tiempo "val" }
```

Código Fuente No. 45

Práctica 3. 2

Rayo de luz. Esta práctica muestra cómo realizar un rayo de luz moviéndose a través de una línea de varios LED. Podremos configurar tanto la velocidad del rayo, así como la longitud.

Al final parecerá como si un rayo de luz sólido atravesase los LED.

Esquema:

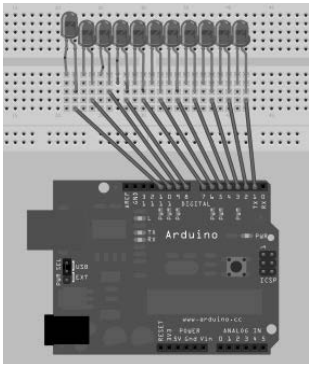

Figura No 159. Configuración de placa de Arduino.

En nuestro IDE de Arduino tendremos:

```
int pinArray [] = { 1,2,3,4,5,6,7,8,9,10,11 };
int controlLed = 13; // LED de control
int waitNextLed = 100; // Tiempo antes de encender el siguiente
LED
int tailLength = 4; // Número de LED que permanecen encendidos
antes de empezar a apagarlos para formar la cola.
int lineSize = 11;
// Número de LED conectados (que es también el tamaño del array)
void setup() // Configuración de los pines como salida digital
```

```
{
int i;
pinMode (controlLed, OUTPUT);
for (i=0; i< lineSize; i++)
{
pinMode(pinArray[i], OUTPUT);
}
}
void loop()
{
int i;
int tailCounter = tailLength; // Se establece la longitud de la
cola en un contador
digitalWrite(controlLed, HIGH); // Se enciende el LED de control
para indicar el inicio del loop
for (i=0; i<lineSize; i++)
{
digitalWrite(pinArray[i],HIGH); // Se encienden consecutivamente
los LED
delay(waitNextLed); // Esta variable de tiempo controla la
velocidad a la que se mueve el rayo de luz
if (tailCounter == 0)
{
digitalWrite(pinArray[i-tailLength],LOW); // Se apagan los LED en
función de la longitud de la cola.
}
else
if (tailCounter > 0)
tailCounter--;
}
for (i=(lineSize-tailLength); i<lineSize; i++)
```

Código Fuente No. 46

Bibliografia

Jorge Pomares Baeza. *Manual de Arduino.* Universidad de Alicante. Disponible en: http://rua.ua.es/dspace/bitstream/10045/11833/1/arduino.pdf

José Antonio E. García Álvarez. *Resistencia Eléctrica.* Disponible en: http://www.asifunciona.com/electrotecnia/ke_resistencia/ke_resistencia_1.htm

Luis González. I.E.S. *Sistemas de Numeración.* Santa Eugenia. Disponible en: http://platea.pntic.mec.es/~lgonzale/tic/binarios/numeracion.html