ARDUINO

PARA ARTISTAS

RODRIGO S. A.

EVENTOS, ILUMINACIÓN,

CINÉTICA Y PANTALLAS

Importante

PRÓLOGO

Las artes plásticas utilizan elementos susceptibles de ser moldeados, modificados y convertidos por el artista. Sus recursos plásticos son la materia prima para su expresión, perspectiva o punto de vista.

Las nuevas generaciones de artistas plásticos al sentir su expresividad limitada inmediatamente consideran incorporar tecnologías a sus obras, ya sean de pintura, dibujo, escultura, grabado, cerámica, orfebrería o artesanía. Claro que no es sencillo sumar cinética, audio, interactividad y conectividad a las artes plásticas, pero quienes se animan obtienen resultados asombrosos. Quizás por la calidez de los recursos originales potenciados por recursos tecnológicos o porque los límites sencillamente se extienden en esta nueva sinergia

¿Pero cuál es el camino a seguir? ¿Es necesario aprender programación? ¿Ser técnico electrónico? ¿Saber soldar? ¿Saber modelado 3D? ¿Y qué clase de hardware conviene utilizar? ¿Placas Arduino, placas ESP o placas Raspberry? ¿Se consiguen los recursos necesarios con facilidad? ¿Son caros?

A menudo, los artistas se desaniman cuando intentan bucear en la inmensidad de recursos multidisciplinarios; y sus obras, ya con la inquietud de potenciarlas, moverlas, hacerlas sonar, conectarlas a las redes quedan estáticas y disminuidas.

Este libro tiene como fin intentar llenar este vacío y hacerlo de una manera amena, clara y entretenida.

Arduino para Artistas es un libro eminentemente práctico y bajo la idea rectora de "aprender haciendo". ¿Esto quiere decir que no hay teoría? En absoluto. La teoría desde luego está incluida, pero de tal manera que no es un pre-requisito u obstáculo , sino un extra que permite fundar las bases para llevar todos los mini proyectos un poco más allá.

La organización de capítulos es completamente funcional. Es decir que, tras unas explicaciones iniciales para conseguir materiales, partes y aplicaciones, se pasa directamente a la realización de proyectos que permiten por ejemplo realizar movimientos, tocar sonidos, encender luces y también acciones más complejas como conectarse a Internet y tomar datos.

Es posible que tengas pensado para tu obra cierta funcionalidad y las probabilidades son que la vas a encontrar explicada, pero también puede suceder que veas lo sencillo que es hacer tal o cual cosa y que el libro funcione a la inversa, como un motor de ideas.

¿Qué tipo de funcionalidades están explicadas en el libro para sumar a tus obras de arte?

- Cómo encender luces: pequeñas luces de electrónica, tiras led, lámparas incandescentes.

- Cómo incorporar sonidos, voces y música.

- Cómo mover partes: motores DC, motores servo y motores paso a paso.

- Cómo detectar eventos: distancia, presión, presencia, sonido y redes WiFi.

- Cómo presentar información en pantallas: lcd, teles de tubo, oled y hasta tinta digital.

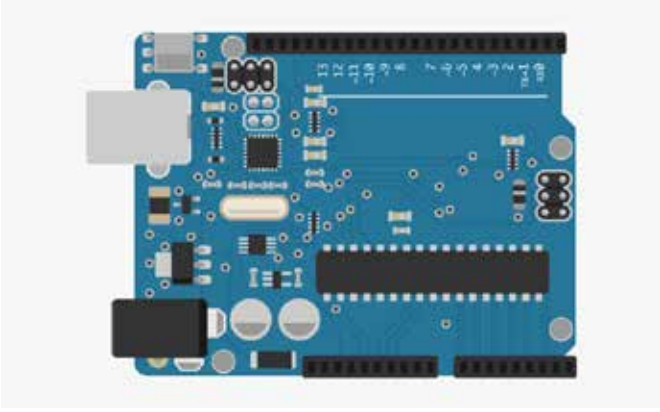
- Cómo conectar la obra a Internet: recibir datos de webs, notificar vía Telegram.

Los capítulos no son bloques de información, sino instrucciones concretas para llevar a cabo los pequeños proyectos que realicen estas funcionalidades específicas. Cada mini proyecto se presenta acompañado de la listas de materiales con sus links– muchos de ellos se repiten –, y código que puede ser copiado si no hay interés en aprender a programar.

¿Listo para empezar?

QUÉ ES ARDUINO

El cerebro, por llamarlo de alguna forma, de todos los proyectos contenidos en este libro es algo llamado Arduino. ¿Pero qué es Arduino exactamente?



Arduino es una plataforma electrónica de código abierto basada en hardware y software fácil de utilizar. Las placas Arduino permiten leer entradas como por ejemplo un dedo presionando un interruptor o bien la humedad del ambiente

y convertir esto en una salida como por ejemplo, encender una luz o publicar un Tweet. Para indicarle al Arduino qué hacer se le otorgan una serie de instrucciones en una variante de lenguaje de programación C++ al microcontrolador de la placa.

El proyecto Arduino comenzó en el año 2003 como un programa para estudiantes en... Italia ¿Quién lo hubiera dicho? No debería extrañar tanto dado que Italia es un país con muchísimos logros en todas las áreas, incluyendo artes, diseño e industria. El nombre de hecho proviene de un bar en Ivrea, Italia, donde se juntaban los fundadores.

La placa original es fabricada por Smart Projects en Italia, pero también hay modelos lanzados por SparkFun Electronics, Adafruit Industries y muchas factorías chinas que producen compatibles bastante confiables.

El hecho de haber lanzado Arduino bajo licencia Open Source logró que esta placa fuera el cen-

tro de numerosos proyectos de todo tipo: desde hobbys personales, hasta instalaciones artísticas y complejos sistemas científicos Asimismo logró una competencia de precios tal que hoy en día es muy, pero muy económico adquirir placas Arduino.

TIPOS DE PLACAS ARDUINO

Las placas Arduino más comunes son:

ARDUINO UNO

Es el punto de entrada a Arduino. Por su tamaño es fácil de manipular. No hace falta soldar y trae conectores muy sencillos. Trae memoria flash de 32KB, EEPROM de 1KB y 2KB de SRAM. Tiene 14 pines de entrada/salida digitales, 6 entradas analógicas y una línea de comunicación serial.

ARDUINO MEGA

La principal diferencia son las características técnicas. Esta placa trae 256kb de memoria flash programable. 8KB de RAM y 4KB de SRAM. Por parte de los pins trae 54 pins digitales que pueden ser utilizados como entrada o salida. 16 pins análogos y 4 líneas de comunicación serial.

ARDUINO NANO

Es más económica y chica que el Arduino Uno. Trae 14 pines digitales y 6 analógicos. Una de sus versiones con ATmega168 tiene 0.512KB EEPROM, 1KB SRAM y 16kb Flash. La otra con ATmega328P tiene 1KB EEPROM, 2KB SRAM y 32kb Flash.

ARDUINO LILYPAD

Tiene un diseño circular plano y se suele utilizar para crear dispositivos que "se visten" Trae procesador ATmega168V, 6 entradas análogicas. 14 IO 6 PWM digitales. 0.512KB EEPROM, 1 SRAM, 16KB Flash.

ARDUINO PRO MICRO

Es una placa muy pequeña. Trae procesador ATmega32U4 con interface HID para simular operaciones de teclado y mouse. 5 PWM pins, 12 entradas/salidas digitales.

ARDUINO MKR WAN 1300

Es una placa original Arduino con Atmel SAMD21 y módulo Murata CMWX1ZZABZ Lo-Ra. La pla-

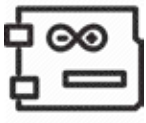
ca puede ser alimentada por 2 pilas AA o AAA o bien entrada de 5V. A diferencia de otros Arduino, esta placa usa 3.3V en los pines. Clock 32.768 kHz (RTC), 48 MHz Quizás su principal función sea la conectividad LORA, que se trata de una red inalámbrica de baja potencia y área amplia (10-20KM) sobre dispositivos de bajo consumo.

ARDUINO MKR ZERO

Tiene procesador SAMD21 Cortex-M0+ 32bit ARM MCU, 256 KB flash, 32 KB SRAM. Soporta batería Li-Po single cell, 3.7V, 700mAh minimum. Es una placa ideal para proyectos de audio y música dado que trae un lector de tarjetas microSD y permite reproducir audio sin hardware adicional por medio de la nueva librería https://arduino.cc/en/Reference/ArduinoSound Asimismo trae incorporado un Real Time Clock para ejecutar funciones en cierta hora o bien grabar la hora en registros.

Luego hay dos placas que no son oficialmente Arduino, pero que se programan de la misma forma y que agregan funciones de conectividad WiFi. Son la ESP8266 y la ESP32.

Para la mayoría de los proyectos de este libro se utiliza Arduino Nano. Algunos proyectos requieren un motor shield y por tal motivo usan Arduino Uno. Otros requieren conectividad y por lo tanto, ESP o Arduino MKR son utilizados.



REQUERIMIENTOS

Como se dijo al final del capítulo anterior, si bien se utiliza la misma placa para la mayoría de los proyectos y de hecho es una placa muy económica y popular, ciertas funcionalidades necesitan otro tipo de placa. Tal es el caso de los proyectos con conectividad a Internet (ESP32) o bien los que utilizan motor shields.

Quienes sepan soldar pueden realizar circuitos más pequeños y óptimos, pero quienes no lo sepan pueden sencillamente montar los circuitos en un breadboard – que es una placa para hacer conexiones con cables cuyos extremos son pines - sin necesidad de soldar.

En cuando a los gabinetes, los artistas plásticos cuentan con enormes recursos para encapsular los circuitos, así que no es una gran preocupación de este libro. Llegado el caso hay otro libro de la editorial llamado Electronic Enclosures for

3d Printing. Este libro enseña a modelar gabinetes en un software gratuito llamado Fusion 360 para imprimir en impresoras o servicios de impresión 3D. Aún no está traducido al español.

Para comenzar recomiendo la siguiente lista de materiales:

- Arduino Nano – no hace falta la placa Arduino original, puede ser un clone
- Jumper cables hembra-hembra
- 1 Cable micro USB
- 1 Computadora PC, Mac o Linux (para programar el Arduino)
- 1 Led
- 1 Servo motor SG90
- 1 ficha micro plugin

Con estos elementos ya es posible encender luces, reproducir sonidos y mover partes. Llegado

el caso luego se pueden conseguir requerimientos específicos de ciertos proyectos. Ejemplo: el sensor para detectar presencia o bien la placa Arduino MKR para conectividad.

PROYECTOS

Además de los capítulos con teoría respectiva a la placa Arduino y a cómo instalar el software, los mini proyectos contenidos y explicados son:

Iluminación: se explica cómo encender y apagar un led, cómo encender y apagar una tira de leds, cómo encender y apagar lámparas incandescentes y cómo armar secuencias.

- Sonidos: se explica cómo reproducir bips, melodías, efectos sonoros, voces y música en mp3.

- Detección de eventos: se explica cómo detectar la distancia a un objeto, cómo detectar activación de interruptores, cómo detectar movimiento y presencia de sonido.

- Cinética: cómo mover motores DC,

motores servo y motores paso a paso.

- Presentar información en pantallas: cómo presentar información en pantallas lcd, oled y hasta tinta digital.

- Conexión a Internet: se explica cómo enviar tweets, actualizar páginas, notificar vía Telegram.

MONTAJE

ARMAR LA ELECTRÓNICA

Tras conseguir los requerimientos, la primera parte consiste en armar la electrónica. Ya sea en un breadboard – que es una placa con conectores temporales a fin de evitar soldaduras –, conectando los jumper cables del Arduino a las partes adicionales con jumper cables hembra hembra con cables soldados, recién cuando está todo conectado al Arduino se pasa al paso siguiente que consiste en subir el código.

INSTALAR EL ARDUINO IDE

Para subir el código de programación al Arduino es necesario utilizar un software llamado Arduino IDE. El mismo se descarga gratis desde la web https://www.arduino.cc/en/Main/Software En el caso que tras la edición de este libro la di-

rección hubiera cambiado, basta con realizar una pequeña búsqueda online para averiguar la nueva dirección para la descarga gratuita del software. El software viene en tres versiones: Windows, Linux y MacOs. En el caso de Linux es importante verificar los permisos del puerto USB. En el caso de Windows y MacOs no hay mayores inconvenientes.

CONFIGURAR EL ARDUINO IDE

Los dos aspectos a configurar son: el tipo de placa Arduino y el puerto USB a utilizar. Tras conectar el Arduino a la computadora, el IDE va a mostrar los puertos disponibles en la pestaña Herramientas/Puerto. Basta con ir ahí y seleccionar el Puerto al que se conectó el cable USB. Si hay varios, ir probando de a uno. En cuanto al tipo de placa, esta se selecciona desde Herramientas/Placa.

Para ESP8266 y ESP32 es necesario realizar un paso adicional: ir a Archivos, Preferencias, Dirección adicional del administrador de placas. Puede ser que esta leyenda varíe, pero es fácil de ubicar. Ahí se copia y pega esta dirección

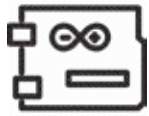
http://arduino.esp8266.com/stable/package_esp8266com_index.json y luego aparecerá en Herramientas/Placa NodeMCU.

Para Arduino MKRZero hace falta ir al Boards Manager y descargar SAM Boards.

COPIAR EL CÓDIGO AL ARDUINO IDE

Si bien es posible tipear cada línea de código de cada proyecto en la pestaña de código del Arduino IDE, lo mejor es descargar los archivos .ino y no tener que tipear. Por otro lado se descartan errores de tipeo dado que los archivos a descargar están probados.

Para obtener los archivos de cortesía de este libro basta con escribir un comentario del libro en Amazon y cuando el mismo sea publicado enviar un email a la editorial steadmanthompson@gmail.com Por favor comunicarse solo cuando el comentario haya sido publicado.



DEPURACIÓN DE ERRORES

Si bien el código de estos proyectos fue probado, es posible que se produzca algún error ya sea de tipeo o bien al intentar modificar los proyectos. En cualquier caso es bueno saber que Arduino puede ir dando mensajes a la computadora a través de una comunicación serial. Todos los proyectos tienen código para ir informando por serie el punto de ejecución y la forma de visualizar esto es ir al Arduino IDE, Herramientas, Monitor Serial. Luego es necesario comprobar que la velocidad indicada en el código sea la misma que la indicada en el selector desplegable del Arduino IDE y luego irán apareciendo los mensajes que entrega el Arduino cuando se ejecuta el código. En el ejemplo que sigue debajo, la velocidad es 9600 y al ser ejecutado imprimirá "Arduino iniciado" en la ventana de monitor serial.

```
Serial.begin(9600);
Serial.print("Arduino iniciado");
```

AGREGADO DE LIBRERÍAS

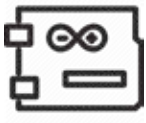
Ciertos proyectos necesitan librerías externas que deben ser descargadas de Internet e instaladas en la computadora donde reside el Arduino IDE. Un ejemplo de código que necesitaría una librería es la línea

```
#include <LibreriaABC.h>
```

Si al enviar el código al Arduino, el IDE indica que no puede encontrar LibreriaABC.h, es necesario descargarla de Internet. Esto se hace sencillamente desde el menu Sketch, Include Library, Manage Library.

Si la librería no se encuentra ahí es posible bajarla manualmente de Internet e instalarla agregando el ZIP desde Sketch, Add Zip. Sino también se puede descomprimir el zip, y copiar.

```
En Windows a: Documents\Arduino\
libraries.
En MAC a: Documents/Arduino/
libraries
```



HARDWARE DEL ARDUINO

Este capítulo es teórico y puede ser pasado por alto hasta que sea necesario comprender mejor el hardware del Arduino.

Las patas del microcontrolador – por ejemplo el ATMega328 – se conectan a unos pines llamados header sockets. Estos se dividen en tres grupos principales: pines digitales, pines análogos y pines de corriente.

Estos pines transfieren voltaje y son importantes porque permiten conectar fácilmente circuitos, sensores y otros dispositivos al Arduino para crear proyectos de todo tipo.

PINES DIGITALES

Como su nombre lo indica, permiten enviar o recibir señales digitales. Es decir que admiten uno de dos estados: encendido o apagado, traducidos en 0 o 5V (o bien 0 o 3.3v en ciertas placas)

PINES ANALÓGICOS DE ENTRADA

Se usan para recibir valores analógicos entre 0 y 5v. El valor concreto puede ser cualquiera en el medio: 0.1, 2.3v, etc

PINES ANALÓGICOS DE SALIDA

Si bien Arduino parece no tener pines de salida analógicos, en realidad sí los tiene, pero disfrazados en pines digitales marcados como PWM. En algunas placas figuran con el símbolo ~ PWM quiere decir pulse-width modulation. PWM es una técnica para dar la impresión de una salida analógica utilizando pines digitales. La técnica es sencilla: si bien la salida solo puede tomar 0/1, se hacen cambios muy rápidos entre estos estados. Si la salida está prendida el 50% del tiempo, tiene 50 de duty cycle. Esto represen-

ta el porcentaje de tiempo en que la salida está activa y puede tener cualquier valor porcentual, por ejemplo 20%. Si se aplica a un led, el led va a tener menor potencia porque el ojo humano llega a captar así esos cambios porcentuales. Con un motor, la velocidad de rotación va a ser más lenta.

PINES DE CORRIENTE

VIN quiere decir Voltage In y se puede utilizar para darle corriente al Arduino, GND quiere decir ground o tierra. 5v quiere decir salida de 5v y 3V salida de 3.3v para potenciar circuitos externos.

LEDS DEL ARDUINO

Los leds estándar del Arduino son los siguientes. Led verde: led de encendido. Led de RX y TX: datos transmitidos hacia y desde el Arduino. L es el Led interno, conectado al pin13, pero puede ser otro pin dependiendo de la placa utilizada.

Los proyectos de sonido del libro utilizan pines digitales para emitir audio. En el caso de los buzzers, cualquier pin puede ser configurado para este fin. En el caso de los proyectos que usan librería Mozzi, el pin D9 es utilizado y por lo tanto no debe ser conectado a otros dispositivos.

PROGRAMACIÓN

PROGRAMACIÓN DEL ARDUINO

Nota: este capítulo puede ser pasado por alto en el caso que solo te interese llevar adelante los proyectos del libro tal como están. Ahora bien, si te interesa aprender a modificar los proyectos es imperativo conocer algo sobre la programación del Arduino.

PARTES DE UN CÓDIGO ARDUINO

En principio, cualquier código Arduino consta de dos partes: setup y loop. La primera parte solo se ejecuta una vez y como su nombre lo indica contiene el "setup" o la configuración inicial. La segunda parte se ejecuta repetitivamente o bien en "loop".

Esto es importante para determinar adónde colocar qué partes de nuestro código. En el pri-

mero de los proyectos por ejemplo se utilizan las líneas previas a la función setup para definir variables globales – que van a ser requeridas en todo el código – luego se usa la sección setup para definir la función de los pines, cuáles están en uso y si son de entrada o salida y finalmente se usa la sección loop para ir prendiendo y apagando las luces. Estas instrucciones se ejecutan secuencialmente y vuelven a iniciarse.

¿CÓMO SE DEFINE UNA VARIABLE?

```
int ledVerde=2;
```

Esto quiere decir que ledVerde es de tipo Integer y se le asigna el valor 2. Todas las líneas en el código Arduino tienen que finalizar con punto y coma.

¿CÓMO SE DEFINE UN PIN?

```
pinMode(ledVerde, OUTPUT);
```

Con pinMode se indica primero el número de pin y luego si el pin es de entrada o salida. El número de pin puede indicarse directamente en la

función - pinMode(2,OUTPUT); - pero conviene tenerlo definido antes, para poder cambiarlo en un lugar para todo el código posterior.

Los pines son conectores del Arduino cuyo fin es adosar luces, sensores y dispositivos externos varios. En un caso sencillo como el proyecto del semáforo, al pin 2 se conecta un extremo del led y el otro extremo del led se conecta al pin GROUND o tierra. El pin entonces es un pin de salida o OUTPUT dado que Arduino enviará un voltaje y no estará leyendo nada.

En otro caso como la conexión de un interruptor, un extrema del mismo puede ir conectado al pin 2 y el otro a Ground. La definición en este caso será de INPUT dado que en lugar de enviar una señal, se recibe y lee una señal.

¿CÓMO SE INSERTAN COMENTARIOS?

Utilizando dos barras iniciales se definen comentarios en el código, lo cual es ideal para documentar lo que se hace. Por ejemplo:

```
// inicialización de los pines
```

Para comentar muchas líneas se utiliza

```
    /*
Todas las líneas estas
estarán comentadas en el código
*/
```

¿CÓMO SE INSERTA UNA LÍNEA PARA DEPURAR CÓDIGO?

La función genérica de depuración ya está explicada anteriormente, pero el código necesario para obtener en la computadora información sobre lo que está pasando en Arduino es la siguiente:

```
Serial.begin(115200);
Serial.println("Iniciado");
Serial.println("Contador:
"+contador);
```

La primera línea indica que se abre una línea de comunicación serial a 115200 de velocidad. La segunda línea enviará desde el Arduino al Arduino IDE (software instalado en la PC, Mac o Li-

nux), la leyenda Futbol. La depuración puede ser utilizada para entender qué sector del código se está utilizando o bien para imprimir el valor de las variables o la temperatura de un sensor, por ejemplo. La tercera línea imprime el valor de la variable contador-

CONDICIONALES

Los condicionales son sectores del código que solo se ejecutan cuando se cumple una condición. En Arduino IDE, el modelo del condicional sigue a continuación. Es importante notar que la comparación de igualdad se hace con DOS símbolos de igual, para diferenciarlo de la asignación. La condición en el ejemplo es: cuando el estado del Relay sea HIGH, cambiarlo a LOW y viceversa.

```
if (relayState==HIGH){
    digitalWrite(relay, LOW);
  }
else{
   digitalWrite(relay, HIGH);
  }
```

LOOPS

Independientemente de que el código principal de Arduino se ejecute en Loop una y otra vez, quizás uno necesita adentro del código que otro sector se ejecute en loop y esto se hace con la instrucción while y una condición especificada entre paréntesis.

```
while(relayState==HIGH){
      // aquí el código a repetir
}
```

FOR

En el caso que sea necesario repetir un segmento de código iterando valores de un contador, es posible usar el siguiente código

```
 for(int i=1; i<=10; i++){
    // aquí el código que se
ejecuta de 1 a 10
}
```

FUNCIONES

Las funciones son ideales para reusar partes de código. En Arduino IDE se definen anteponiendo function y el retorno de la función se hace con la palabra clave return. En el ejemplo que sigue a continuación se muestra una función que reemplaza el símbolo de grado por la palabra "grados"

```
function removeSpecial($str){
    $str=str_replace("°","
grados ",$str);
    return $str;
}
```

Valores al azar

Para darle cierto jugo a los sonidos a veces es necesario invocar valores al azar que luego se pueden aplicar por ejemplo a un vibrato. Primero se llama randomSeed(analogRead(0)); considerando que la entrada del pin 0 no sea utilizada. El ruido va a causar que distintos números se usen como semilla para generar el número al azar. El ejemplo que figura debajo imprime un número al azar entre 0 y 10.

```
long randNumber;

void setup() {
  Serial.begin(9600);

  randomSeed(analogRead(0));
}

void loop() {
  randNumber = random(10);
  Serial.println(randNumber);

  delay(50);
}
```

El Led Integrado

En las etapas de prototipos de los proyectos con Arduino es muy común utilizar un led para comprobar funciones, por ejemplo: la administración remota vía un web server local en NodeMCU. En lugar de conectar un breadboard con un led y una resistencia, es posible utilizar un led interno que tiene la placa sin necesidad de recurrir a todas las molestias externas. ¿Cómo se usa

el Led interno? Dado que cada placa tiene dicho Led en otro pin, es necesario hacer referencia a la constante LED_BUILTIN.

Para definir

```
pinMode(LED_BUILTIN, OUTPUT);
```

Para encender

```
digitalWrite(LED_BUILTIN, HIGH);
```

Para apagar

```
digitalWrite(LED_BUILTIN, LOW);
```

Almacenar datos permanentes con EEProm

No muchos lo saben pero Arduino permite almacenar datos en una memoria permanente como si fuera un pequeño disco rígido, sin necesidad de utilizar un lector de tarjetas SD externo. Obviamente que su capacidad es muy limitada, pero para ciertos proyectos es más que suficiente.

Cada byte de la memoria EEPROM puede manejar un valor entre 0 y 255.

```
#include <EEPROM.h>
```

```
int addr = 0;

void setup() {

}

void loop() {

  int val = 100;


  EEPROM.write(addr, val);

// se pasa a la celda siguiente
de la memoria y si llega al tope,
vuelve a 0

// para leer  EEPROM.read(addr);

  addr = addr + 1;
  if (addr == EEPROM.length()) {
    addr = 0;
  }

  delay(100);
```

```
}
```

ILUMINACIÓN

ENCENDER Y APAGAR UN LED

El primer proyecto consiste en encender y apagar un Led. No hace falta alimentación adicional para el Led dado que la alimentación del mismo Arduino es suficiente. Para conectar el Led es necesario pelar dos cables y pegar con cinta los extremos a las patas del Led. Mirando a trasluz el Led es posible determinar que el segmento más grande es el que va conectado a GND

PARTES

- Arduino Nano https://amzn.to/3867RWR
- Led https://amzn.to/3sM6AMw

Circuito: Conectar una para del Led a GND y la

otra pata del Led al pin D2.

```
void setup() {
   pinMode(2, OUTPUT); // acá se
le indica al Arduino que vamos a
conectar el Led al pin digital 2
}

void loop() {
   digitalWrite(2, HIGH);     //
enciende
  delay(500);                 //
espera medio segundo
   digitalWrite(2, LOW);      //
apaga
  delay(500);                 //
espera medio segundo
}
```

LÁMPARAS INCANDESCENTES

Encender una lámpara incandescente requiere otro acercamiento ya que el voltaje involucrado es mayor - 220 V versus 5V con los que opera el Arduino - ¿Qué se hace entonces? El Arduino se utiliza para activar un dispositivo llamado Relay,

que es una especie de corte. Para entenderlo intuitivamente: se conecta la lámpara a 220V y siempre está encendida. Ahora se corta un cable y se pone en el medio el Relay. Con una señal de programación le decimos al Relay que conecte nuevamente ambos cables, la corriente vuelve a fluir y la lámpara se enciende.

PARTES

- Arduino Nano https://amzn.to/3867RWR
- Relay 1 canal https://amzn.to/3bdWBcA

CIRCUITO

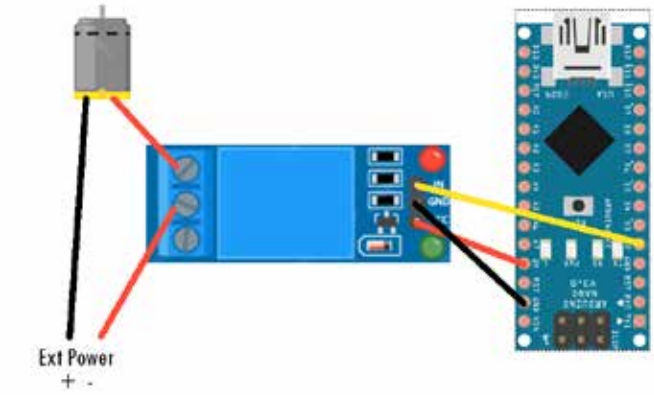
Ahora el pin D2 será conectado al Relay junto a GND y VCC. Luego un cable de la lampara será cortado al medio y conectado a los pines NO y al pin del medio.

```
void setup() {
  pinMode(2,OUTPUT);
}
```

```
void loop() {
  digitalWrite(2,HIGH);
  delay(500);
  digitalWrite(2,LOW);

}
```

Nota: el mismo método puede ser utilizado para encender muchas otros dispositivos alimentados por 220V.



Cuidado al manipular los cables de 220V. Hacerlo en un domicilio con disyuntor y sin tocar cables que se encuentren conectados.

ENCENDER Y APAGAR UNA TIRA DE LEDS

Alimentar una tira de Leds como las que se venden en casas de iluminación requiere el mismo acercamiento anterior, pero en lugar de alimentar desde 220V se hace desde la fuente que viene con las luces Led. Para tal fin el cable que se corta es el que va del transformador a la tira de luces.

PARTES

- Arduino Nano https://amzn.to/3867RWR
- Relay 1 canal https://amzn.to/3bdWBcA

CIRCUITO

Ahora el pin D2 será conectado al Relay junto a

GND y VCC. Luego un cable de la lampara será cortado al medio y conectado a los pines NO y al pin del medio.

```
// Arduino para artistas
// Editorial Steadman Thompson
//Licencia                https://
creativecommons.org/licenses/
by/4.0
void setup() {
  pinMode(2,OUTPUT);
}

void loop() {
  digitalWrite(2,HIGH);
  delay(500);
  digitalWrite(2,LOW);

}
```

ARMAR SECUENCIAS

El armado de secuencias se logra iterando encendidos y apagados y puede involucrar más de una luz. A modo de ejemplo, el siguiente código enciende gradualmente tres lámparas.

```
// Arduino para artistas
// Editorial Steadman Thompson
//Licencia                https://
creativecommons.org/licenses/
by/4.0
void setup() {
  pinMode(2,OUTPUT);
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
}

void loop() {
  digitalWrite(2,HIGH);
  delay(500);
  digitalWrite(2,LOW);

  digitalWrite(3,HIGH);
  delay(500);
  digitalWrite(3,LOW);

  digitalWrite(4,HIGH);
  delay(500);
  digitalWrite(4,LOW);

}
```

Es posible también utilizar la sentencia while para lograr iteraciones más complejas:

```
var = 0;
while (var < 50) {
  // enciende y apaga Led en pin
2 50 veces
 digitalWrite(2,HIGH);
  delay(500);
  digitalWrite(2,LOW);
  var++;
}

var = 0;
while (var < 25) {
  // enciende y apaga Led en pin
3 25 veces
 digitalWrite(3,HIGH);
  delay(500);
  digitalWrite(3,LOW);
  var++;
}
```

ÍNDICE

Arduino para artistas
Autor Rodrigo S. A.
Editorial Steadman Thompson