12

PULSADOR

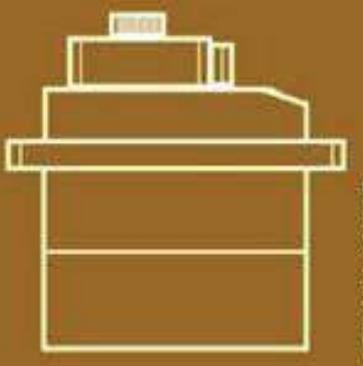
DIODO LED

RESISTENCIA DE 10 KILO OHMIOS

RESISTENCIA DE 220 OHMIOS

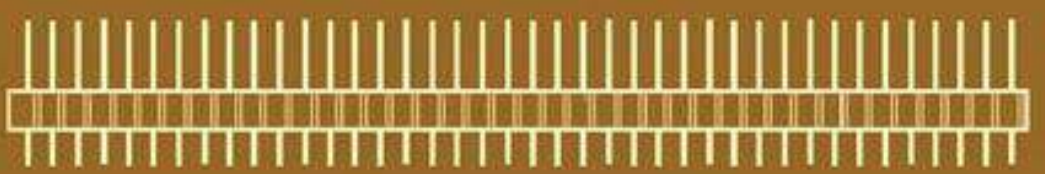
RESISTENCIA DE 1MEGA OHMIO

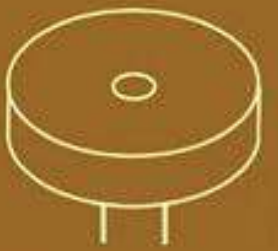
CONDENSADOR DE 100uF



SERVO MOTOR

TIRA DE PINS MACHO (SOLO SON NECESARIOS 3 PINS)





ZUMBADOR PIEZOELÉCTRICO

INGREDIENTES

Traducido by Tino Fernández Cueto

http://www.futureworkss.com

MECANISMO DE BLOQUEO SECRETO

CONSTRUIR UN MECANISMO DE BLOQUEO SECRETO PARA EVITAR QUE ALGUIEN PUEDA SUSTRAER LAS COSAS QUE TENGA GUARDADAS EN UNA CAJA

Descubra: entrada con un zumbador, escribir sus propias funciones

Tiempo: 1HORA

Nivel: alto

Proyectos en los que se basa: 1,2,3,4,5

El zumbador usado para reproducir sonidos en los proyectos del theremin y del teclado también se puede usar como un dispositivo de entrada o sensor. Cuando se le aplica una tensión de 5V, este sensor puede detectar las vibraciones las cuales pueden ser leídas por las entradas analógicas de Arduino. Es necesario conectar una resistencia de un valor alto (del orden de 1Mega ohmio) entre la salida y masa para que funcione correctamente, es decir, se realiza el montaje del zumbador en serie con una resistencia de 1Mega ohmio conectada a masa, de manera que el conjunto forma un divisor de tensión.

Cuando un zumbador piezoeléctrico es presionado contra una superficie plana puede vibrar, como por ejemplo sobre una mesa de madera, siendo Arduino capaz de detectar la intensidad de esta vibración. Usando esta información puede comprobar si el número de vibraciones detectadas se encuentran dentro del rango de trabajo establecido. En el código puede contabilizar el número de vibraciones y ver si coinciden con el número de vibraciones almacenadas.

Un pulsador permite bloquear el motor en una posición. Algunos diodos LEDs aportan información: un LED rojo indicará que la caja está bloqueada, un LED verde indicará que la caja no está bloqueada, y un LED amarillo permite saber si se ha recibido el número correcto de vibraciones.

También escribirá su propia función la cual le permitirá saber si una vibración es demasiado fuerte o demasiado débil. El escribir sus propias funciones le ayuda a ahorrar tiempo de programación al llamar siempre a una misma parte del código sin necesidad de volverlo a escribirlo cada vez que se use. Las funciones pueden usar argumentos y devolver valores a través de estos argumentos. En este caso, utilizará una función para determinar el nivel de volumen de la vibración. Si se encuentra dentro del rango correcto se incrementará el valor de una variable.

Es posible construir el circuito simplemente por hacerlo, con ninguna finalidad, pero es mucho más interesante y didáctico hacerlo con un objetivo, en este caso realizar una caja de seguridad en donde guardar algo. Si dispone de una caja de madera o una caja de cartón puede realizar una serie de agujeros en su interior, usar un servomotor para abrir y cerrar un pestillo, y así evitar que cualquiera pueda abrir la caja para coger lo que hay en su interior.

MONTANDO EL CIRCUITO

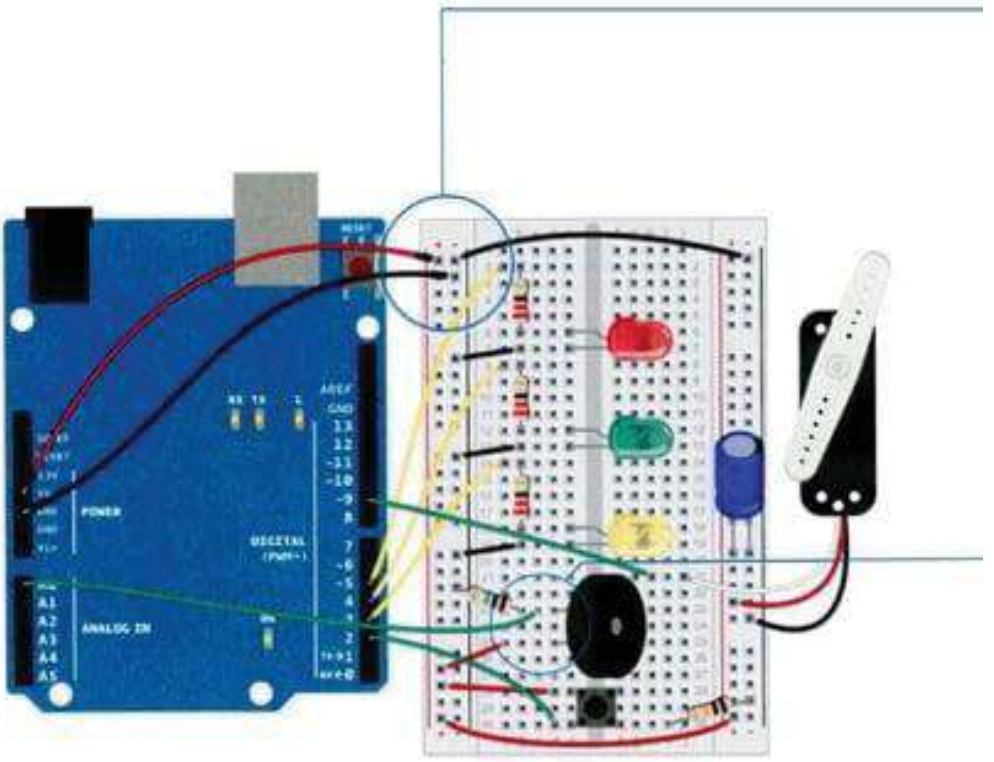


Figura 1

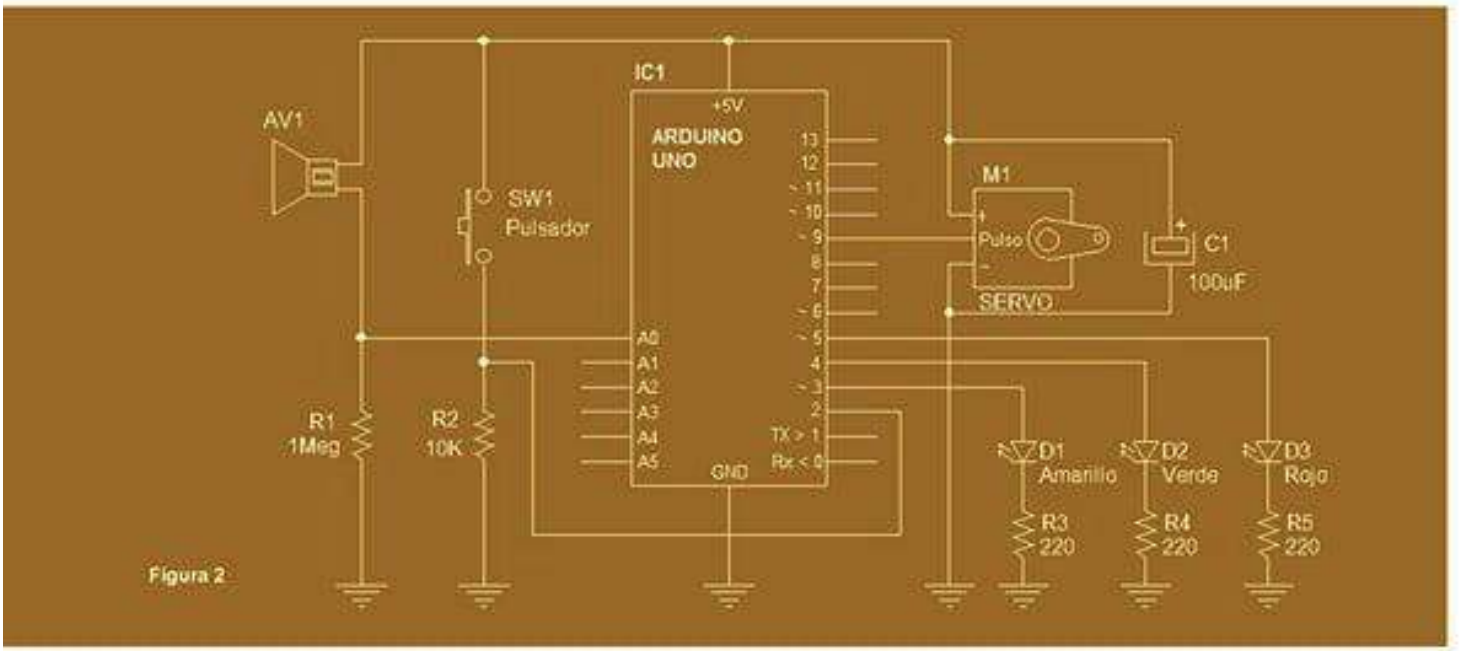


Figura 2

Hay muchas conexiones en la placa, asegurarse que todas estas conexiones están realizadas correctamente.

1. Conectar los cables de alimentación y masa a ambos lados de la placa de pruebas. Colocar el pulsador sobre la placa de pruebas conectando uno de sus terminales a +5V. El otro extremo del pulsador conectarlo a masa a través de una resistencia de 10Kilo ohmios. Conectar la unión de la resistencia y el terminal del pulsador al pin digital número 2 de Arduino

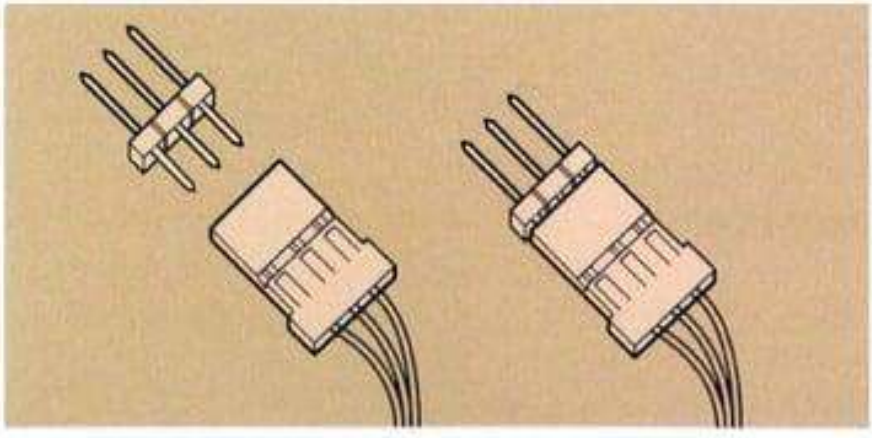
2. Colocar el zumbador sobre la placa de pruebas y a continuación conectar un cable desde uno de sus terminales al positivo de la alimentación. Si el zumbador tiene un cable de color rojo o el símbolo "+", este será el terminal que se conecte al positivo. Si en el zumbador no se indica la polaridad se podrá conectar de cualquier forma. Conectar el otro terminal del zumbador al terminal Analog 0 de Arduino. Colocar una resistencia de 1Mega ohmio entre masa y este terminal conectado a Arduino. Valores de resistencia más bajos harán que el zumbador sea menos sensible a las vibraciones.

3. Cablear los LEDs, conectando los cátodos (patilla corta) a masa, y colocar un resistencia de 220 ohmios en serie con los ánodos. A través de sus respectivas resistencias, conectar el LED amarillo al pin digital 3 de Arduino, el LED verde al pin digital 4, y el LED rojo al pin digital 5.

4. Insertar el conector macho de tres terminales dentro del conector hembra del servomotor (ver figura 3). Conectar el cable rojo a la alimentación y el cable negro a masa. Colocar un condensador electrolítico de 100 micro faradios en las líneas de alimentación de la placa de pruebas para suavizar las variaciones bruscas de corriente y evitar variaciones de tensión, además asegurarse que el condensador se ha conectado con la polaridad correcta. Conectar el terminal de datos del servomotor, el cable blanco, al terminal 9 de Arduino.

El servomotor viene con un conector hembra, así que será necesario añadirle un conector macho con tres terminales para poder conectarlo a la placa de pruebas

Figura 3



EL CÓDIGO

Librería Servo

Como en el proyecto del "Indicador de estado de ánimo", es necesario importar la librería **Servo** y crear una copia para usar el motor.

Constantes útiles

Crear constantes para nombrar las entradas y las salidas

Variables para guardar los valores del pulsador y del zumbador

Crear variables para guardar los valores del pulsador y del zumbador

Umbrales de los golpes

Configurar algunas variables de tipo constante para usar como límites para los niveles máximo y mínimo de los golpes

Variables para el estado del bloqueo y el número de golpes

La variable bloqueado permite saber si el bloqueo está activado o no. Una variable de tipo **boolean** contiene datos que solo pueden ser verdadero – true (1), o falso – false (0). Cuando se aplica tensión al circuito debe de empezar con el mecanismo desbloqueado. La última variable de tipo global guardará el número de golpes válidos que haya recibido.

Configurar la dirección de los pines digitales e inicialización del servo y del puerto serie

Dentro de la función **setup()**, conectar el servo al pin 9.
Establecer los pines de los LEDs como salidas y los pines del pulsador como entradas.

Desbloquear

Inicializar la comunicación serie con el ordenador para poder ver el nivel de los golpes, en que estado se encuentra el bloqueo y el número de golpes válidos que se han efectuado. Encender el diodo LED verde, mover el servo a la posición de desbloqueo, y mostrar el estado actual en el monitor serie indicando que el circuito está en la posición de desbloqueo.

Comprobar el pulsador

Dentro de **loop()**, primero se comprueba si la caja está bloqueada o no lo está. Esto determinará lo que ocurra en el resto del programa. Si está bloqueado se procede a leer el estado del pulsador.

```
#include <Servo.h>
Servo miServo;

const int zumbador = A0;
const int PinPulsador = 2;
const int LedAmarillo = 3;
const int LedVerde = 4;
const int LedRojo = 5;

int ValorGolpe;
int ValorPulsador;

const int GolpesSuaves = 10;
const int GolpesFuertes = 100;

boolean bloqueado = false;
int NumeroGolpes = 0;


void setup() {
 miServo.attach(9);
 pinMode(LedAmarillo, OUTPUT);
 pinMode(LedRojo, OUTPUT);
 pinMode(LedVerde, OUTPUT);
 pinMode(PinPulsador, INPUT);
 Serial.begin(9600);

 digitalWrite(LedVerde, HIGH);
 miServo.write(0);
 Serial.println("La caja esta desbloqueada!");
}


void loop() {
 if(bloqueado == false){
  ValorPulsador = digitalRead(PinPulsador);
```

Bloqueado

Si el pulsador está cerrado (está presionado), cambiar el valor de la variable bloqueado a true (verdadero), indicando que el bloqueo está activado. Apagar el diodo LED verde y encender el LED rojo. Es necesario activar el monitor serie para que aparezca la indicación "La caja está bloqueada!" y de esta manera ver el estado del bloqueo. Se mueve el servo a la posición de bloqueo. Se añade un tiempo de 1 segundo para que el bloqueo tenga tiempo de realizarse.

Leer los golpes en el sensor

Si la variable bloqueo es true, y el bloqueo está activado, leer el valor de la vibración del zumbador y almacenarlo en **ValorGolpe.**

Contar solo los golpes válidos

Las siguientes instrucciones comprueban si se han efectuado menos de tres golpes válidos y si se percibe alguna vibración en el sensor (zumbador). Si ambas condiciones son ciertas, comprueba si el golpe en este momento es válido o no lo es, e incrementa la variable **NumeroGolpes** en caso de ser válido. En esta parte del código es donde se realiza la llamada a la función personalizada **VerificarGolpes()**. Se escribe esta función al final de la función **loop()**. **VerificarGolpes()** comprueba que el valor de la variable **ValorGolpe** es un golpe válido o no lo es, además esta variable forma parte del argumento de esta función. Después de que se ejecute esta función se indica, a través del monitor serie, el número de golpes válidos que faltan para realizar el desbloqueo.

Desbloquear

Comprobar si se han efectuado tres o más golpes válidos. Si esta condición se cumple, cambiar la variable de bloqueo a false (falso), y mover el servomotor a la posición de desbloqueo. Esperar 20 mili segundos para que le de tiempo a mover el servomotor, y cambiar el estado del LED verde a encendido y apagar el LED rojo. Mostrar un mensaje en el monitor serie indicando que "La caja está desbloqueada!".

Cerrar la instrucción **else** y la función **loop()** con un parte de llaves.

Definir una función para comprobar la valided de los golpes

Ahora es el momento de escribir la función personalizada **VerificarGolpes()**. Cuando se escriben funciones personalizadas es necesario indicar si va a devolver un valor o no. Si no va a devolver un valor la función se declara del tipo void, igual que las funciones **loop()** o **setup()**. Si va a devolver un valor se debe de declarar de que tipo (**int, long, float**, etc). En este caso se comprueba si un golpe es válido (true) o no lo es (false) por eso se declara la función de tipo boolean.

```
28      if(ValorPulsador == HIGH){
29        bloqueado = true;
30        digitalWrite(LedVerde, LOW);
31        digitalWrite(LedRojo, HIGH);
32        miServo.write(90);
33        Serial.println("La caja esta bloqueada!");
34        delay(1000);
35       }
36      }
```

```
37      if(bloqueado == true){
38       ValorGolpe = analogRead(zumbador);
```

```
39       if(NumeroGolpes < 3 && ValorGolpe > 0){
40        if(VerificarGolpes(ValorGolpe) == true){
41         NumeroGolpes++;
42        }
43        Serial.print(3-NumeroGolpes);
44        Serial.println(" golpes para abrir");
45       }
```

```
46      if(NumeroGolpes >= 3){
47        bloqueado = false;
48        miServo.write(0);
49        delay(20);
50        digitalWrite(LedVerde, HIGH);
51        digitalWrite(LedRojo, LOW);
52        Serial.println("La caja esta desbloqueada!");
53        NumeroGolpes = 0;
54        }
55       }
56      }
```

```
57      boolean VerificarGolpes(int valor){
```

La misión de esta función será controlar un número (que contiene la variable **ValorGolpe**) para ver si es válida o no. Para pasar el valor de esta variable dentro de la función se crea una variable de tipo entero (**int valor)** dentro del argumento en el momento de crear dicha función.

Comprobar la valided del golpe

En esta función, cada vez se refiere a la variable **Valor** se usará cualquier número que contenga la variable **ValorGolpe** dentro del programa principal y que se transmite a través del argumento de la función. En esta línea del programa se comprueba que el dato que contiene la variable **Valor** es mayor que el dato que contiene la variable **GolpesSuaves** y menor que el dato que contiene la variable **GolpesFuertes**. Por ejemplo, si el dato de **ValorGolpe** vale 25, se pasa a la función a través del argumento con lo cual la variable **Valor** tendrá este dato y a continuación se comprueba que es mayor que 10 y menor de 100, según se puede ver en la línea adjunta de la página siguiente.

Indicando que el golpe es válido

Si el dato de la variable **Valor** se encuentra entre los dos valores anteriores (entre 10 y 100) entonces existe un golpe válido, por tanto el diodo LED amarillo se enciende durante 50 milli segundos y aparece un texto en el monitor serie indicando que se ha producido un golpe válido "Golpe válido de valor " así como dicho valor.

Función retorno válido

Para permitir al programa principal usar el resultado de la comparación que se acaba de realizar dentro de la función se utiliza el comando **return** (retorno). Este comando **return** también hace que la función finalice cuando es usado: una vez que se ejecuta la función, retorna al programa principal.

Indicando golpe no válido. Función retorno no válido

Si el dato de la variable **Valor** está por debajo del dato de la variable **GolpesSuaves** (10) o por encima de **GolpesFuertes**(10) aparece el siguiente texto en el monitor serie "El valor de golpe no es válido" y se retorna al programa principal a través de return false.

Cerrar la función personalizado usando más de una llave.

COMO SE UTILIZA

Cuando se conecta Arduino a un ordenador y se carga el programa en él, a continuación hay que abrir el monitor serie. Debe de ver como el diodo LED verde se enciende y el servomotor se mueve a la posición de desbloqueo.

En la ventana del monitor serie aparece la frase "La caja esta desbloqueada!".

Ahora para probar el circuito presionar el pulsador. El diodo LED verde se apaga y se enciende el LED rojo y se muestra la frase "La caja esta bloqueada!" en el monitor serie.

Deberá de determinar el nivel de los golpes que consiguen desbloquear el mecanismo efectuando varios golpes sobre el zumbador con diferentes niveles de intensidad. En el momento que el diodo LED amarillo se encienda durante menos de 1 segundo y aparezca la frase "Golpe valido de valor x" sabrá la intensidad de los golpes para desbloquear el mecanismo.

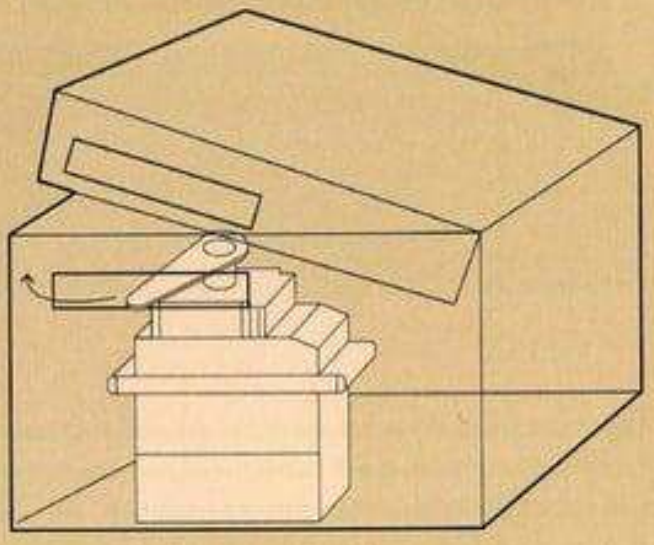
```
58    if(valor > GolpesSuaves && valor < GolpesFuertes){
59      digitalWrite(LedAmarillo, HIGH);
60      delay(50);
61      digitalWrite(LedAmarillo, LOW);
62      Serial.print("Golpe valido de valor ");
63      Serial.println(valor);
64      return true;
65     }
66    else {
67      Serial.print("El valor del golpe no es valido ");
68      Serial.println(valor);
69      return false;
70     }
71  }
```

También en el monitor serie se muestra el valor del golpe válido así como el número de golpes que es necesario efectuar para producir el desbloqueo.

El número de golpes válidos siempre serán 3 y cuando se consigan efectuar estos 3 golpes sobre el zumbador el diodo LED rojo se apagará, el LED verde se encenderá y el servomotor se moverá 90 grados a la posición de desbloqueo, además de aparecer una notificación en el monitor serie indicando que el bloqueo está desactivado.

El valor para el golpe ideal puede variar con respecto a los que aparecen en este ejemplo. Esto depende de un número de variables diferentes, como el tipo de superficie sobre la que se apoya el sensor, si el zumbador está sólidamente sujeto o se mueve cada vez que se golpea, etc. Usando el monitor serie y golpeando varias veces el zumbador se podrá determinar el valor más apropiado de la intensidad del golpe cada vez que se encienda el diodo led amarillo. También se puede hacer usando uno de los ejemplos que incluye el IDE de Arduino y que se localiza dentro "**Ejemplos**", "**Analog**" y el programa "**AnalogInOutSerial**". Puede acceder a la siguiente página web para ver en más detalle como usar este programa: ***arduino.cc/analogtoserial***

Si se coloca este proyecto dentro de una caja, será necesario hacerle unos agujeros para los LEDs y el pulsador. También será necesario colocar le una palanca al servomotor que permita bloquear la tapa de la caja. Es necesario hacer otro agujero para pasar el cable USB que se conecta al ordenador y así poder ver la intensidad de los golpes que permiten abrirla.
También será necesario reorganizar la colocación de la placa de pruebas y de la tarjeta Arduino, o soldar los LEDS y el pulsador mediante unos cables de forma que se puedan montar en la caja. El soldar consiste en unir dos o más metales, normalmente un cable y un terminal de un componente, usando otro material que permite fijarlos permanentemente y que no permita que se suelten a pesar de que se produzca cualquier movimiento brusco. Sin nunca ha soldado antes, es mejor preguntarle a alguien que ya tenga experiencia en este tipo de trabajos para que pueda ayudarle, o intentar practicar con algunos cables que tenga a mano antes de hacerlo con algunos de los componentes de este proyecto. Cuando se suelda algo esto significa que la conexión realizada al soldar es permanente, así que asegurarse después de realizar la conexión de los LEDs y el pulsador que los cables no se sueltan con facilidad, simplemente puede tirar de ellos para ver si la soldadura está bien realizada. Dirigirse a la página ***arduino.cc/soldering*** para ver una buena explicación de cómo soldar.



❶

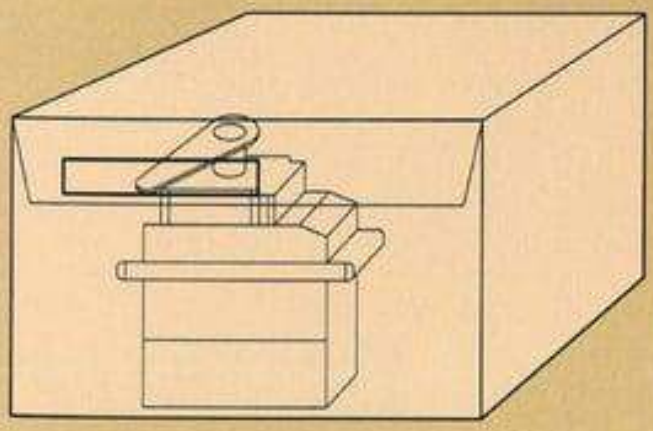
Hacer dos agujeros en la caja: uno en una de las paredes de la base y el otro en un lateral de la tapa. Colocar el servo en el interior de manera que la pestaña se pueda mover del interior al exterior de los agujeros cuando se bloquea.

Escribir sus propias funciones no solo permite controlar el flujo del código con más facilidad, también ayuda a que los proyectos sean más legibles aunque el código se vaya haciendo cada vez más y más grande. Con el tiempo, a medida que escriba más código, podrá disponer de un gran número de funciones que pueda usar en diferentes proyectos, haciendo este proceso más rápido y único por su estilo de programar. (Nota del traductor) Debe de gustarle la programación sino será imposible el hacerlo.

En este proyecto simplemente se cuenta el número de golpes válidos (3) para desbloquear el mecanismo, no importa el tiempo que se tarda en conseguirlo. Se puede mejorar el proyecto al usar un temporizador con la instrucción **mills()**. Usar el monitor serie para comprobar si los golpes válidos se efectúan en un periodo de tiempo específico. Puede volver a mirar el Proyecto del Reloj de Arena Digital (página 86) para ver como trabaja esta instrucción. No está limitado a encontrar los golpes válidos en un rango de tiempo. Puede buscar patrones complejos de golpes sobre la base de la cantidad de vibración y tiempo juntos. Existe un gran número de ejemplos en Internet que tratan de como hacer esto, buscar "**Arduino knock lock**" para encontrar más ejemplos de este tipo de proyecto.

En la página https://www.arduino.cc/en/Tutorial/KnockSensor puede obtener más información de como se usar el zumbador como un sensor de golpes

Los zumbadores de tipo piezo eléctrico se pueden usar como entradas cuando se conectan formando un divisor de tensión junto con una resistencia de gran valor. El diseño de una función es una forma fácil de escribir código que se puede usar para tareas concretas y/o en otros proyectos diferentes.



❷

Fijar el servo en el interior con un poco de pegamento, asegurandose de nuevo que la pestaña del servo puede girar facilmente a través de las ranuras