|  |
| --- |
| Software para Robots Jordán Pascual : pascualjordan@uniovi.es |

|  |  |
| --- | --- |
| 2 | Sensores y actuadores |

# Introducción

El objetivo principal de esta práctica es utilizar las entradas y salidas de Arduino para controlar otro tipo de elementos más complejos (sensores, actuadores y elementos de interfaz de usuario).

Los **sensores** nos permiten obtener información del entorno. Para utilizar un sensor desde Arduino se suele utilizar al menos una entrada digital o analógica dependiendo del tipo de sensor. Algunos sensores pueden requerir el uso de varias salidas, o de varias entradas para enviar parámetros que configuren su funcionamiento.

La mayor parte de **actuadores** se manejan mediante el uso de salidas, los elementos de interfaz de usuario (tales como: pantallas, teclados, botones, etc.), suelen utilizar entradas y/o salidas dependiendo del modelo concreto y de las capacidades funcionales que tenga.

# Ejemplos

**Detección de luz**

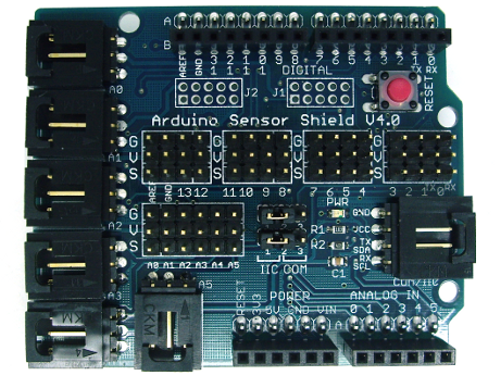
En este ejemplo vamos a utilizar un sensor de luz para monitorizar el nivel de luz de una habitación, en caso de que el nivel de luz sea bajo encenderemos un led.

|  |  |
| --- | --- |
|  | **Sensor de luz:** este tipo de sensores deben conectarse a una salida analógica, retorna un valor que suele oscilar entre 0 - 620 , a mayor valor retornado mayor cantidad de luz está detectando.  En este modelo el pin derecho va a GND, el pin central a V5, y el pin izquierdo a una entrada analógica (por ejemplo A2). |
|  | **Diodo LED:** Un diodo, es un componente electrónico que solo permite pasar la corriente en una dirección. En la dirección del positivo al negativo (la parte ancha del triángulo). La pata positiva del LED es la larga. |

**Construcción del circuito :**

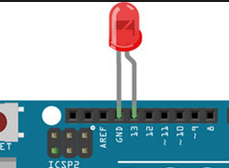
1. **Opción 1**: Protoboatd + Arduino: Conectamos el sensor de luz a: **GND (negro), 5V (Rojo) y A2 (azul).**

**Opción 2**: Sensor shield . La forma de conexión más sencilla es utilizar una de las columnas de la placa de sensores:



1. Conectamos directamente sobre la placa un led al Pin 13 y GND. El pin 13 incluye una resistencia por lo que no hace falta colocar una.

Realmente el pin 13 ya incluye un pequeño led (justo debajo), pero no obstante conectamos nuestro led para darle mayor visibilidad.



**Programación de Arduino:**

1. Inicializamos el puerto para registrar la salida en el método setup().  
   También activamos el pin digital 13 como salida.

Serial**.**begin**(**9600**);**

pinMode**(**13**,** OUTPUT**);**

1. En cada interacción del método loop() obtenemos el valor del sensor de luz leyendo directamente la entrada analógica.

int valorLuz **=** analogRead**(**A2**);**

Serial**.**println**(**"Valor: "**+**String**(**valorLuz**));**

1. Evaluamos el valor retornado por el sensor en caso de que sea bajo encendemos el led.

int valorLuz **=** analogRead**(**A2**);**

Serial**.**println**(**"Valor: "**+**String**(**valorLuz**));**

**if** **(** valorLuz **<** 200**){**

digitalWrite**(**13**,** HIGH**);**

**}** **else** **{**

digitalWrite**(**13**,** LOW**);**

**}**

1. Comprobamos el funcionamiento del programa , podemos tapar la fotorresistencia con la mano para obtener valores luz bajos.

|  |
| --- |
| void setup**()**  **{**  // No hace falta declarar las lecturas analógicas  Serial**.**begin**(**9600**);**  pinMode**(**13**,** OUTPUT**);**  **}**  void loop**()**  **{**  // leer lectura analógica  int valorLuz **=** analogRead**(**A2**);**  Serial**.**println**(**"Valor: "**+**String**(**valorLuz**));**    **if** **(** valorLuz **<** 200**){**  digitalWrite**(**13**,** HIGH**);**  **}** **else** **{**  digitalWrite**(**13**,** LOW**);**  **}**  **}** |

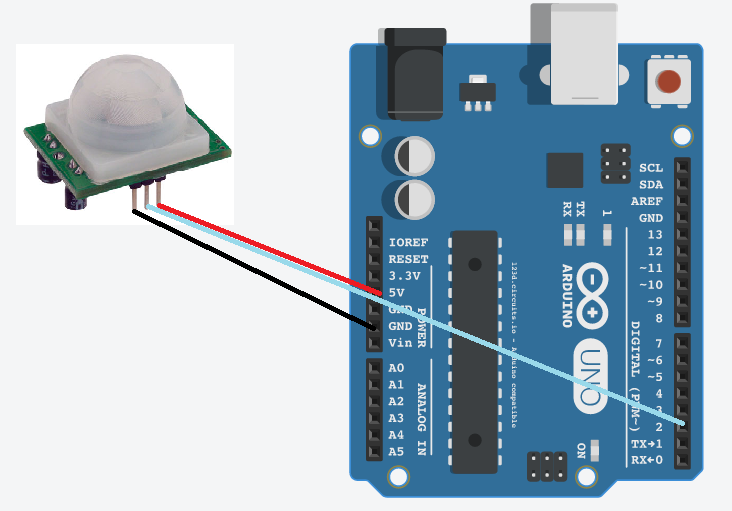
**Detección de movimiento**

En este ejemplo utilizamos el sensor de movimiento de infrarrojos para detectar movimiento cerca del sensor.

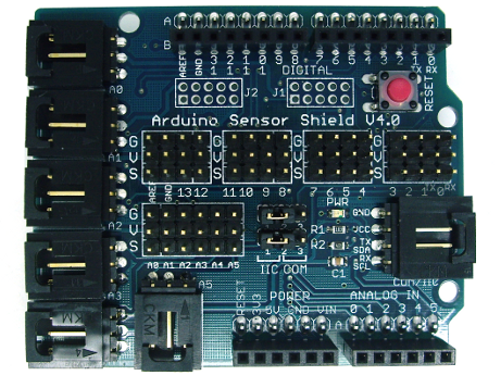
|  |  |
| --- | --- |
| http://hetpro-store.com/TUTORIALES/wp-content/uploads/2015/02/pir1.jpg | **Sensor Pir:** permite detectar movimiento cercano.  El sensor está recubierto por una lente de fresnel que le ayuda a incrementar el área de funcionamiento.  Cuando detecta movimiento mantiene una salida de 5V, cuando no hay movimiento mantiene una salida de 0V (algunas marcas funcionan de la forma contraria) . Para detectar movimiento el sensor utiliza un detector de niveles de radiación infrarroja, la mayor parte de cosas que se mueven generan un pequeño nivel de radiación infrarroja (calor).  Dependiendo del tipo de sensor puede tener diferente distancia de detección y diferente Angulo de detección, unos valores típicos para un sensor de este tamaño son 7 metros y 110º.  Algunos sensores como este incluyen cierta capacidad de calibración, en este caso tenemos dos tornillos en la parte inferior, que nos permiten ajustar el grado de sensibilidad del sensor y el retardo entre detecciones. |

**Construcción del circuito :**

1. Conectamos el sensor PIR a: **GND, 2 y 5V**



**Opción 2**: Sensor shield . La forma de conexión más sencilla es utilizar una de las columnas de la placa de sensores:



1. Conectamos directamente sobre la placa un led al Pin 13 y GND. El pin 13 incluye una resistencia por lo que no hace falta colocar una.

**Programación de Arduino:**

1. Inicializamos el puerto para registrar la salida en el método setup().  
   También activamos el pin digital 2 como entrada.

Serial**.**begin**(**9600**);**

pinMode**(**pinsensor**,** INPUT**);**

1. En cada iteración del método loop() obtenemos el valor del sensor leyendo la entrada digital.

int pirValor **=** digitalRead**(**pinsensor**);**

Serial**.**println**(**"Valor: "**+**String**(**pirValor**));**

1. Evaluamos el valor retornado y una pequeña pausa entre mediciones

**if(**pirValor **==** HIGH**){**

Serial**.**println**(**"Detectado movimiento"**);**

**}** **else** **{**

Serial**.**println**(**"- - - -"**);**

**}**

|  |
| --- |
| int pinsensor **=** 2**;**  void setup**(){**  Serial**.**begin**(**9600**);**  pinMode**(**pinsensor**,** INPUT**);**  **}**  void loop**(){**  int pirValor **=** digitalRead**(**pinsensor**);**  Serial**.**println**(**"Valor: "**+**String**(**pirValor**));**    **if(**pirValor **==** HIGH**){**  Serial**.**println**(**"Detectado movimiento"**);**  **}** **else** **{**  Serial**.**println**(**"- - - -"**);**  **}**    delay**(**50**);**  **}** |

## Prueba otros sensores digitales

|  |  |
| --- | --- |
|  | **Sensor de vibración SW-400:** permite detectar vibraciones por encima de un umbral (el umbral se puede configurar de forma física ajustando el tornillo).  Cuando no detecta vibración retorna una salida de 0V (LOW), cuando la detecta retorna 5V (HIGH).  La patilla **D0** se conecta a un pin digital, la VCC a un voltaje y la patilla central a **GND.**  **Ajuste:** si giramos el tornillo hasta su límite en sentido anti horario, lo colocamos en su mayor nivel de sensibilidad (detecta vibraciones muy leves). Cuanto más giremos el tornillo de forma horaria más reduciremos la sensibilidad y más fuertes deberán ser las vibraciones para ser detectadas. |

Probar el funcionamiento del sensor de vibración, se puede utilizar el mismo código que para el ejemplo anterior.

|  |  |
| --- | --- |
| Resultado de imagen de arduino sensor proximidad infrarrojo | **Sensor de proximidad por infrarrojos:** permite detectar elementos colocados delante del sensor (la distancia de detección se puede configurar de forma física ajustando el tornillo). Está compuesto por un emisor de infrarrojos que emite pulsos de forma continua, y un detector de infrarrojos, cuando la luz infrarroja rebota contra un objeto el detector capta ese rebote (de esa forma sabemos que hay algo delante).  Cuando no detecta ningún elemento retorna una salida de 5V (HIGH), cuando detecta un elemento retorna 0V (LOW).  La patilla **OUT** se conecta a un pin digital, la VCC a un voltaje y la patilla GND a una tierra (G) del Arduino).  **Ajuste:** si giramos el tornillo hasta su límite en sentido anti horario, lo colocamos en su menor nivel de detección (nunca detecta nada)  Cuanto más giremos el tornillo de forma horaria más aumentaremos el nivel de detección (CM), primero detectará elementos que estén únicamente a 0 - 2 cm, según vallamos girando el tornillo podrá llegar a detectar a 0 - 8 cm (aproximadamente) |

## Control de distancia mínima

En este ejemplo utilizamos el sensor de ultrasonidos para detectar la distancia a la que colocamos un objeto.

|  |  |  |
| --- | --- | --- |
| http://www.kurtsik.org/blog/wp-content/uploads/HC-SR04.png | **Sensor de ultrasonidos SR04 :** detectar la distancia a la que se encuentra un objeto.  Este sensor emite y recibe el rebote de un ultrasonido, calculando el tiempo transcurrido entre estas dos acciones es posible saber si la onda colisiono con algún objeto.    Dependiendo del fabricante y el tipo del sensor pueden tener más o menos precisión y alcance: esta versión permite medir distancias de 3cm - 3m con una precisión aproximada de 3mm.  El sensor se conecta a **VCC y GND**. Además, necesita otros dos pines digitales, **Trig** se conectará a un ping digital y se encargará de enviar el pulso ultrasonido, **Echo** se conectará a otro pin digital y se encargará de recibir el eco de dicho pulso. | |
| **En ocasiones el sensor de sonidos deja de emitir mediciones o solo emite mediciones erróneas en ese caso volver a desconectar el USB y volver a conectarlo.** | |

**Construcción del circuito :**

1. Conectamos el sensor de ultrasonido a: **V5, Pin digital 9 (Trig), Pin digital 8 (Echo) y GND**

**Programación de Arduino:**

1. Creamos las variables globales que utilizaremos a lo largo de todo el programa.

long distancia**;**

long tiempoRespuesta**;**

int pinTrig **=** 9**;**

int pinEcho **=** 8**;**

1. Inicializamos el puerto para registrar la salida en el método setup().  
   También activamos los 2 pins, uno como salida y otro como entrada.

Serial**.**begin**(**9600**);**

pinMode**(**pinTrig **,** OUTPUT**);** /\* trig envía el pulso ultrasónico \*/

pinMode**(**pinEcho**,** INPUT**);** /\* Echo capta el rebote del pulso

ultrasónico\*/

1. En cada interacción del método loop() colocamos el pin Trig a LOW y realizamos una breve pausa.

digitalWrite**(**pinTrig**,**LOW**);** /\* Por seguridad volvemos a poner el Trig a LOW\*/

delayMicroseconds**(**5**);**

Emitimos la señal HIGH por el ping Trig y realizamos otra breve pausa.

digitalWrite**(**pinTrig**,** HIGH**);** /\* Emitimos el pulso ultrasónico \*/

delayMicroseconds**(**10**);**

Ya podemos leer el resultado del pin Echo, para ello utilizamos el método especial pulseIn(). Este método nos da los microsegundos que tarda el PIN en cambiar su estado HIGH / LOW.

<https://www.arduino.cc/en/Reference/pulseIn>

tiempoRespuesta **=** pulseIn**(**pinEcho**,** HIGH**);** /\* Medimos la longitud del pulso entrante Cuanto tiempo tarda la entrada en pasar de HIGH a LOW retorna microsegundos \*/  
  
Serial**.**println**(**"Tiempo "**+** String**(**tiempoRespuesta**)+**" microsegundos"**);**

1. Una vez tenemos los microsegundos que tarda el ultrasonido en impactar contra el objeto y volver podemos calcular la distancia.

La velocidad del ultrasonido es 0,034 cm/microsegundos, .pero como solo nos interesa lo que tarda en llegar, podemos dividir el tiempo de respuesta o la velocidad entre 2 (0,017).

Finalmente añadimos una pequeña espera (quizá sea mejor probar con esperas mas grandes).

distancia**=** int**(**0.017**\***tiempoRespuesta**);** /\* Calcular la distancia

conociendo la velocidad \*/

Serial**.**println**(**"Distancia "**+** String**(**distancia**)+**"cm"**);**

delay**(**100**);**

|  |
| --- |
| long distancia**;**  long tiempoRespuesta**;**  int pinTrig **=** 9**;**  int pinEcho **=** 8**;**  void setup**(){**  Serial**.**begin**(**9600**);**  pinMode**(**pinTrig **,** OUTPUT**);** /\* trig envía el pulso ultrasónico \*/  pinMode**(**pinEcho**,** INPUT**);** /\* Echo capta el rebote del pulso  ultrasónico\*/  **}**  void loop**(){**  digitalWrite**(**pinTrig**,**LOW**);** /\* Por seguridad volvemos a poner el Trig  a LOW\*/  delayMicroseconds**(**5**);**    digitalWrite**(**pinTrig**,** HIGH**);** /\* Emitimos el pulso ultrasónico \*/  delayMicroseconds**(**10**);**    tiempoRespuesta **=** pulseIn**(**pinEcho**,** HIGH**);** /\* Medimos la longitud del  pulso entrante Cuanto tiempo tarda la entrada en pasar de HIGH a LOW  retorna microsegundos \*/  Serial**.**println**(**"Tiempo "**+** String**(**tiempoRespuesta**)+**" microsegundos"**);**    distancia**=** int**(**0.017**\***tiempoRespuesta**);** /\* Calcular la distancia  conociendo la velocidad \*/  Serial**.**println**(**"Distancia "**+** String**(**distancia**)+**"cm"**);**  delay**(**100**);**  **}** |

|  |
| --- |
| Algunos modelos de sensor ultrasónico introducen demasiado "ruido", en estos casos puede ser una buena estrategia realizar un conjunto de mediciones muy seguidas y tomar como válida la mediana.  [http://playground.arduino.cc/Main/Average](http://playground.arduino.cc/Main/Average%20)  Debemos tener en cuenta que dependiendo de la orientación la onda puede rebotar sin retornar al lector de ultra sonidos, en ese caso el sensor pensara que no hay nada delante. Es justo por eso por lo que los sensores de ultrasonidos suelen colocarse en servomotores (Sobre todo el robots móviles) , para lanzar la onda en varias direcciones y así tener una mayor posibilidad de éxito. |

## Pantalla de segmentos

En ese ejemplo vamos a utilizar una pantalla de segmentos de cuatro dígitos para notificar la distancia medida por el sensor de ultrasonidos SR04.

|  |  |
| --- | --- |
|  | **Pantalla de 7 segmentos y 4 dígitos:** permite representar valores de cuatro dígitos.  Este componente incluye un chip que permite controlar la pantalla haciendo uso únicamente de una entrada y una salida digital.  Este modelo también permite regular el brillo de la luz emitida.  Las conexiones necesarias son: **CLK (Pin Digital PWR), DIG (Pin Digital), VCC (5V), GND (GND).** |

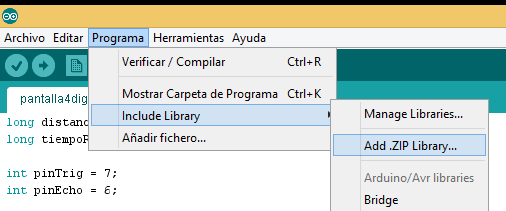
**Construcción del circuito :**

1. Partiendo del circuito anterior conectamos la pantalla a: **CLK (3~), DIG (4), VCC (5V), GND (GND).**

**Programación de Arduino:**

En primer lugar, para poder manejar la pantalla necesitamos descargar e importar una nueva librería: **DigitalTube TM1637**. <https://github.com/reeedstudio/libraries/tree/master/DigitalTube>

Para agregar la librería descargamos el Zip y pulsamos sobre **Programa -> Include Library -> Add. Zip Library**



A partir de este momento la librería estará disponible en la lista de librerías : **Programa -> Include Library -> DigitalTube**

Al incluir la librería veremos que se genera un **#include** al principio de nuestro programa.

#include <TM1637.h>

Vamos a mostrar el valor de un entero en la pantalla.

1. Creamos las variables globales para los pins Clk y Dio.  
   Declaramos un nuevo objeto de tipo TM1637 al cual le indicamos los pins a utilizar.

int pinClk **=** 3**;**

int pinDio **=** 4**;**

TM1637 pantalla**(**pinClk**,**pinDio**);**

1. En el método setup(). inicializamos la pantalla y le establecemos el brillo, posteriormente hay que realizar una pausa para darle tiempo a que se inicie.

El método init() de TM1637 se encarga de iniciar todo lo necesario (incluido las salidas digitales).

pantalla**.**init**();**

pantalla**.**set**(**BRIGHT\_TYPICAL**);**

//BRIGHT\_TYPICAL = 2,BRIGHT\_DARKEST = 0,BRIGHTEST = 7;

delay**(**1500**);**//esperar a que se inicie

1. En el interior del método loop() una vez obtenida la distancia la mostramos por pantalla, para ello usamos el método display(indice de digito, valor[0-9]).

Hay que tener en cuenta que debemos dividir la distancia en dígitos: unidades, decenas, centenas, millares.

int digito0 **=** distancia**/**1000**;**

int digito1 **=** **(**distancia **-** digito0**\***1000**)/**100**;**

int digito2 **=** **(**distancia **-** **(**digito0**\***1000 **+** digito1**\***100**))/**10**;**

int digito3 **=** distancia **-** **(**digito0**\***1000 **+** digito1**\***100 **+** digito2**\***10**);**

pantalla**.**display**(**0**,**digito0**);**

pantalla**.**display**(**1**,**digito1**);**

pantalla**.**display**(**2**,**digito2**);**

pantalla**.**display**(**3**,**digito3**);**

|  |
| --- |
| #include <TM1637.h>  long distancia**;**  int pinClk **=** 3**;**  int pinDio **=** 4**;**  TM1637 pantalla**(**pinClk**,**pinDio**);**  void setup**(){**  Serial**.**begin**(**9600**);**  pantalla**.**init**();**  pantalla**.**set**(**BRIGHT\_TYPICAL**);**  //BRIGHT\_TYPICAL = 2,BRIGHT\_DARKEST = 0,BRIGHTEST = 7;    delay**(**1500**);**//Delay to let system boot  **}**  void loop**(){**  distancia = 1245;    int digito0 **=** distancia**/**1000**;**  Serial**.**println**(**"digito0 "**+** String**(**digito0**));**  int digito1 **=** **(**distancia **-** digito0**\***1000**)/**100**;**  Serial**.**println**(**"digito1 "**+** String**(**digito1**));**  int digito2 **=** **(**distancia **-** **(**digito0**\***1000 **+** digito1**\***100**))/**10**;**  Serial**.**println**(**"digito2 "**+** String**(**digito2**));**  int digito3 **=** distancia **-** **(**digito0**\***1000 **+** digito1**\***100 **+** digito2**\***10**);**  Serial**.**println**(**"digito3 "**+** String**(**digito3**));**    pantalla**.**display**(**0**,**digito0**);**  pantalla**.**display**(**1**,**digito1**);**  pantalla**.**display**(**2**,**digito2**);**  pantalla**.**display**(**3**,**digito3**);**  **}** |

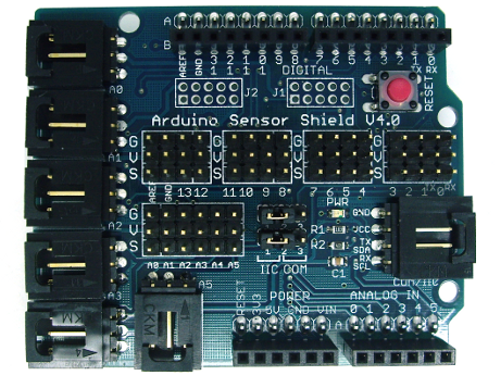
## Servomotor 180°

En este ejemplo vamos a explorar el funcionamiento de un mini servomotor.

|  |  |
| --- | --- |
| http://mlc-s2-p.mlstatic.com/mini-servo-9g-tower-pro-sg90-uc-arduino-pics-pwm-13395-MLC2963192030_072012-O.jpg | **Servomotor 180° :** es un pequeño motor de corriente continua que incluye en su interior una especie de potenciómetro que le permite saber su posición y controlarla.    Gracias a esta particularidad podemos hacer que el motor se mueva a un Angulo exacto, existen servomotores de diferentes tipos: unos permiten rotar 380° (rotación continua) otros 180° o menos, como 60°.  A parte del Angulo de rotación soportado tienen otras características interesantes como el peso que pueden mover o la velocidad a la que se mueven.  El servomotor tiene tres pines:  **Rojo** que se conecta a **5V** (aunque podría ser más)  **Negro** **marrón** que se conecta a **GND**.  **Blanco** o **naranja** que se conecta a un **pin digital.** |

**Construcción del circuito :**

1. Conectamos el servomotor a: **V5 (rojo), Pin digital 8 (naranja) y GND (marrón).**
2. La forma de conexión más sencilla es utilizar una de las columnas de la placa de sensores:



**Programación de Arduino:**

En primer lugar importamos la librería que arduino nos ofrece para controlar el servomotor.

Después de importarla observaremos que se ha incluido un nuevo #include en nuestro programa.

#include <Servo.h>

1. Creamos la variable global de tipo Servo.

Servo servo1**;**

1. Inicializamos el puerto para registrar la salida en el método setup(). asociamos el pin 8 al servo (no hace falta especificar que el pin 8 será una salida digital).

Serial**.**begin**(**9600**);**

servo1**.**attach**(**8**);**

1. En el método loop(). movemos el servo a diferentes grados utilizando el método write(grados) , después de cada movimiento realizamos una pausa.

servo1**.**write**(**180**);**

Serial**.**println**(**"180"**);**

delay**(**3000**);**

servo1**.**write**(**90**);**

Serial**.**println**(**"90"**);**

delay**(**3000**);**

servo1**.**write**(**0**);**

Serial**.**println**(**"0"**);**

delay**(**3000**);**

|  |
| --- |
| #include <Servo.h>  Servo servo1**;**  void setup**()** **{**  Serial**.**begin **(**9600**);**  servo1**.**attach**(**8**);**  **}**  void loop**()** **{**  servo1**.**write**(**180**);**  Serial**.**println**(**"180"**);**  delay**(**3000**);**  servo1**.**write**(**90**);**  Serial**.**println**(**"90"**);**  delay**(**3000**);**  servo1**.**write**(**0**);**  Serial**.**println**(**"0"**);**  delay**(**3000**);**  **}** |

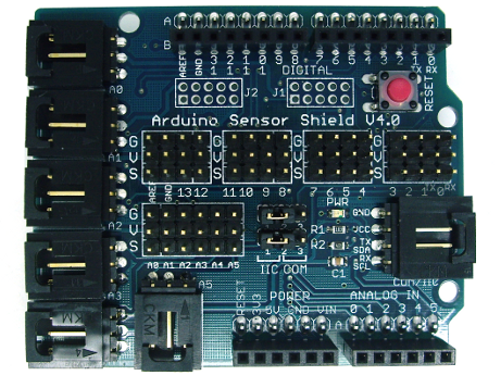
## Led RGB

En este ejemplo vamos a explorar el funcionamiento de un LED RGB.

|  |  |
| --- | --- |
| Resultado de imagen de arduino led RGB | **Led RGB:** circuito integrado con un led RGB, permite alumbrar el Led en 3 colores o realizar combinaciones entre ellos.  El Led tiene 4 pines.  **R** que se conecta a un pin digital (PWM o normal) para controlar el nivel de luz roja  **G** que se conecta a un pin digital (PWM o normal) para controlar el nivel de luz roja  **B** que se conecta a un pin digital (PWM o normal) para controlar el nivel de luz roja  **-** que se conecta a **GND**. |

**Construcción del circuito:**

1. Conectamos las patillas led a: **R** al **S9 , G** al **S10 y B** al **S11 , -** acualquier **G.**



**Programación de Arduino:**

En el siguiente ejemplo controlamos el led con salidas digitales, podemos colocar establecer color (salida) a LOW / HIGH

|  |
| --- |
| int pinRojo **=** 9**;**  int pinVerde **=** 10**;**  int pinAzul **=** 11**;**  void setup**()** **{**  pinMode**(**pinRojo**,** OUTPUT**);**  pinMode**(**pinVerde**,** OUTPUT**);**  pinMode**(**pinAzul**,** OUTPUT**);**  **}**  void loop**()** **{**  digitalWrite**(**pinRojo**,**HIGH**);**  digitalWrite**(**pinVerde**,**LOW**);**  digitalWrite**(**pinAzul**,**LOW**);**  delay**(**500**);**  digitalWrite**(**pinRojo**,**LOW**);**  digitalWrite**(**pinVerde**,**HIGH**);**  digitalWrite**(**pinAzul**,**LOW**);**  delay**(**500**);**  **}** |

En el este ejemplo controlamos el led con salidas analógicas, podemos establecer cada color (salida) a [ 0 – 255].

|  |
| --- |
| int pinRojo **=** 9**;**  int pinVerde **=** 10**;**  int pinAzul **=** 11**;**  int gradoColor **=** 0**;**  void setup**()** **{**  pinMode**(**pinRojo**,** OUTPUT**);**  pinMode**(**pinVerde**,** OUTPUT**);**  pinMode**(**pinAzul**,** OUTPUT**);**  **}**  void loop**()** **{**  analogWrite**(**pinRojo**,** 255 **-** gradoColor**);**  analogWrite**(**pinVerde**,**gradoColor**);**  analogWrite**(**pinAzul**,**0**);**  delay**(**15**);**    gradoColor**++;**  **if** **(** gradoColor **>** 255**)**  gradoColor **=** 0**;**  **}** |

## Teclado

En este ejemplo vamos a utilizar un teclado matricial para introducir el número de grados que debe girar un mini servomotor.

|  |  |
| --- | --- |
|  | **Teclado matricial:** los teclados matriciales permiten capturar pulsaciones en una matriz de botones utilizando el indicador de la fila y la columna pulsada.  Para detectar la pulsación se conectan cuatro pines como entradas a las columnas y cuatro pines como salidas a las filas (o viceversa).  El modo de conexión más común es utilizar 8 pines digitales para conectarlo, pero esto nos dejará sin pines para el resto de elementos.  Nosotros optamos por conectar los cuatro primeros cables (azules) a los pines digitales (2 - 5) y los cuatro cables siguientes (amarillos) a los pines analógicos (A0 - A3). |

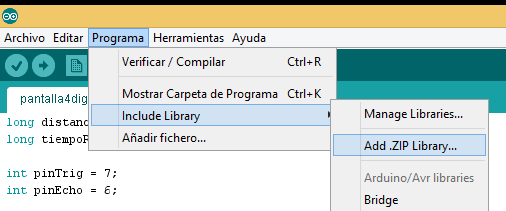
**Construcción del circuito:**

1. Partiendo del circuito anterior conectamos el teclado a: **2 - 5 y A0 - A3** de forma ordenada empezando por el conector de la izquierda.

**Programación de Arduino:**

En primer lugar para poder manejar el teclado necesitamos importar una nueva librería: **KeyPad**. [https://cdn.shopify.com/s/files/1/0557/2945/files/KeyPad\_bb5610fd-0c4c-4b0e-8c5e-1e5b3f83f47f.zip?4187195515480435262](https://cdn.shopify.com/s/files/1/0557/2945/files/KeyPad_bb5610fd-0c4c-4b0e-8c5e-1e5b3f83f47f.zip?4187195515480435262%20)

Para agregar la librería descargamos el Zip y pulsamos sobre **Programa -> Include Library -> Add. Zip Library**



A partir de este momento la librería estará disponible en la lista de librerías : **Programa -> Include Library -> KeyPad**

Al incluir la librería veremos que se genera un **#include** al principio de nuestro programa.

#include <keypad.h>

Continuamos modificando el programa anterior.

1. Creamos las variables globales para el número de filas, columnas y la matriz con todas las teclas que tiene nuestro teclado.

const byte nfilas **=** 4**;**

const byte ncolumnas **=** 4**;**

char teclas**[**nfilas**][**ncolumnas**]** **=** **{**

**{**'1'**,**'2'**,**'3'**,**'A'**},**

**{**'4'**,**'5'**,**'6'**,**'B'**},**

**{**'7'**,**'8'**,**'9'**,**'C'**},**

**{**'\*'**,**'0'**,**'#'**,**'D'**}**

**};**

Creamos dos arrays indicando los pines que vamos a utilizar, para las filas y las columnas.

byte pfilas**[**nfilas**]** **=** **{**2**,**3**,**4**,**5**};** // Filas

byte pcolumnas**[**ncolumnas**]** **=** **{**A0**,**A1**,**A2**,**A3**};** //Columnas

Creamos un objeto de tipo keypad, que recibe, un KeyMap con las teclas, los pins de las filas, los pins de las columnas y el número de filas y columnas.

Keypad teclado **=** Keypad**(** makeKeymap**(**teclas**),** pfilas**,** pcolumnas**,** nfilas**,** ncolumnas **);**

Finalmente añadimos un buffer de lectura en el que iremos almacenando todas las teclas que pulsa el usuario.

1. En el método loop(). pedimos que nos devuelva la tecla obtenida, si el usuario no pulsa nada en cada iteración del loop devolverá vacio '\0'.

Obtenemos la tecla pulsada con el método getKey**();** comprobamos que no es vacía antes de continuar con el procesamiento. En caso de que no sea vacía la mostramos por consola.

char tecla **=** teclado**.**getKey**();**

**if** **(**tecla **!=** '\0'**){**

Serial**.**println**(**"tecla pulsada: "**+**String**(**tecla**));**

...

**}**

Vamos a definir que cuando el usuario pulse la tecla '#' ya ha dejado de escribir. Cualquier tecla distinta a '#' la agregamos al buffer

Cuando el usuario pulsa '#' pasamos todo el contenido del buffer a un entero y lo vaciamos.

**if** **(** tecla **!=** '#'**){**

bufferLectura **=** bufferLectura **+** tecla**;**

Serial**.**println**(**"buffer: "**+**bufferLectura**);**

**}** **else** **{** // es # transformamos a entero

int valorNumerico **=** bufferLectura**.**toInt**();**

Serial**.**println**(**"Aceptado: "**+**String**(**valorNumerico**));**

bufferLectura **=** ""**;** // reinicio

**}**

1. Utilizamos ese "valorNumerico" para indicarle al servomotor la posición que debe tomar.

|  |
| --- |
| #include <Keypad.h>  #include <Servo.h>  Servo servo1**;**  const byte nfilas **=** 4**;**  const byte ncolumnas **=** 4**;**  char teclas**[**nfilas**][**ncolumnas**]** **=** **{**  **{**'1'**,**'2'**,**'3'**,**'A'**},**  **{**'4'**,**'5'**,**'6'**,**'B'**},**  **{**'7'**,**'8'**,**'9'**,**'C'**},**  **{**'\*'**,**'0'**,**'#'**,**'D'**}**  **};**  byte pfilas**[**nfilas**]** **=** **{**2**,**3**,**4**,**5**};** // Filas  byte pcolumnas**[**ncolumnas**]** **=** **{**A0**,**A1**,**A2**,**A3**};** //Columnas  Keypad teclado **=** Keypad**(** makeKeymap**(**teclas**),** pfilas**,** pcolumnas**,** nfilas**,** ncolumnas **);**  String bufferLectura **=** ""**;**  void setup**(){**  Serial**.**begin **(**9600**);**  servo1**.**attach**(**8**);**  Serial**.**println**(**"Setup()"**);**  **}**  void loop**(){**  char tecla **=** teclado**.**getKey**();**  **if** **(**tecla **!=** '\0'**){**  Serial**.**println**(**"tecla pulsada: "**+**String**(**tecla**));**  **if** **(** tecla **!=** '#'**){**  bufferLectura **=** bufferLectura **+** tecla**;**  Serial**.**println**(**"buffer: "**+**bufferLectura**);**  **}** **else** **{** // es # transformamos a entero  int valorNumerico **=** bufferLectura**.**toInt**();**  Serial**.**println**(**"Aceptado: "**+**String**(**valorNumerico**));**  servo1**.**write**(**valorNumerico**);**  bufferLectura **=** ""**;** // reinicio  **}**  **}**  **}** |