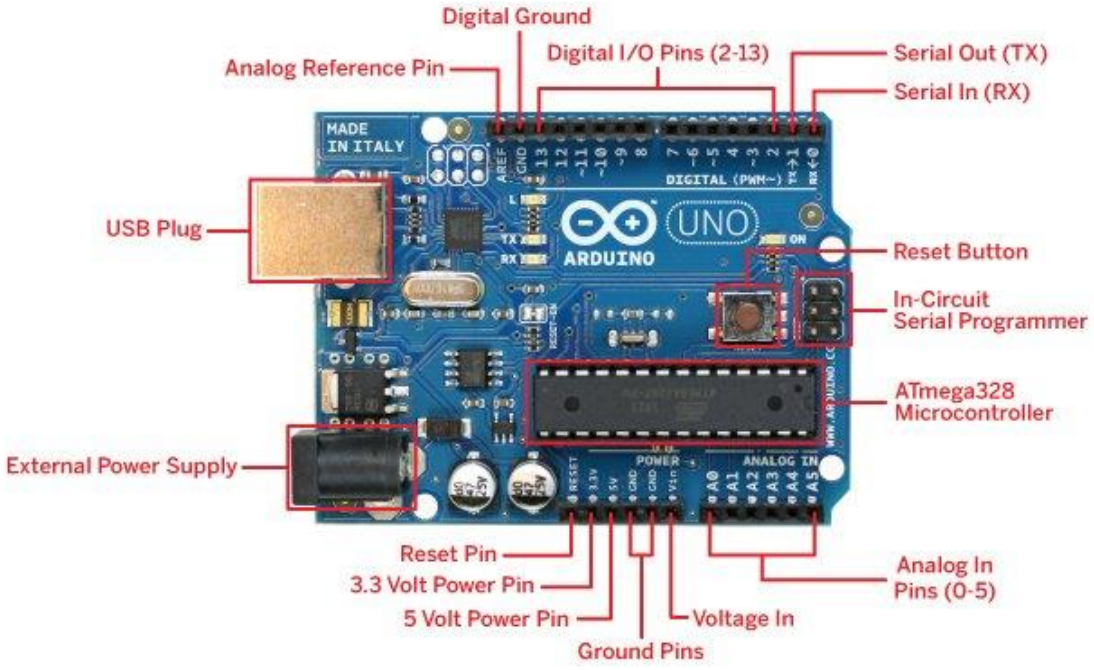
1. Placa Arduino:

Arduino es una tarjeta controladora con la que podemos controlar un sistema electromecánico con un ordenador a través de un lenguaje de programación o código. Es decir, es un pequeño ordenador que es capaz de interpretar la información suministrada por sensores y ejecutar una respuesta, pero al no tener ni teclado ni monitor, necesita de otro ordenador externo donde se introducirá el programa de control, el software. Necesitamos, por tanto, conectar nuestra tarjeta de Arduino con otro ordenador, con un cable USB y con el sistema electromecánico que queremos controlar, lo que haremos mediante una serie de entradas y salidas. Aquí tenéis el esquema:



2. Alimentación

Como todas las tarjetas, es necesario alimentarla eléctricamente. La tarjeta de Arduino se puede alimentar de dos formas:

- Con cable USB: recibirá la alimentación eléctrica del ordenador. Esta alimentación será de 5 V.
- Mediante la clavija de alimentación usando una pila de 9 V o una fuente de alimentación, que debe estar entre 7 y 12 V.
- GND: es la toma de tierra, o nivel 0 V de referencia.

3. Entradas y salidas

La tarjeta cuenta con un grupo de pines numerados del 0 al 13 que pueden actuar como entradas y salidas digitales; algunos de estos pines pueden configurarse como salidas analógicas (3, 5, 6, 9, 10 y 11). Llevan un signo -.
Hay otro grupo de pines numerados desde el A0 al A5, que son entradas analógicas.
En las entradas y salidas digitales, los valores de salida pueden ser 0 V (LOW) o 5 V (HIGH). Se interpreta como entrada con un valor LOW entre 0 y 2 V y como HIGH, entre 3 y 5 V.

En cuanto a las salidas analógicas, los valores de salida oscilan entre 0 y 5 V
La placa cuenta con un led conectado con el pin digital número 13, además del pin de encendido.

4. Software Arduino

El software Arduino IDE Es un entorno sencillo, en el que básicamente tenemos una interfaz para escribir nuestros programas. Tiene una barra de herramientas muy sencilla, que podemos utilizar para verificar el funcionamiento del programa, cargar el programa en la placa, abrir un programa guardado o uno nuevo, guardar el programa actual en disco o abrir una ventana de comunicación.

Debemos tener en cuenta cuando empecemos a utilizar las placas Arduino que hay diferentes modelos de estas placas, por esto es importante asegurarse de que el IDE que estamos usando está bien configurado con nuestra placa. Para ello, hay que ir a Herramientas tarjeta. También hay que comprobar que nuestra tarjeta se encuentra en el puerto correcto, de nuevo hay que ir a Herramientas puerto serial. Una vez comprobados ambos factores podemos empezar a programar.

El circuito electrónico que vamos a controlar con Arduino se montará sobre una placa de protoboard. Es recomendable hacer las conexiones con cables tipo jumper, ya que no se rompen en los zócalos.

5. Nociones básicas

Los programas de Arduino tienen una estructura básica:

Setup(): es una función que solo se ejecuta al principio. Sirve para configurar los pines como entradas (inputs) o salidas (outputs), por ejemplo.

Loop(): esta parte se ejecutará indefinidamente, mientras tenga alimentación eléctrica, claro está. Los comandos incluidos aquí se realizarán por orden, secuencialmente, hasta llegar al último, momento en el que el programa volverá a comenzar.

En Arduino, al programar debemos tener en cuenta:

1. Las funciones comienzan con una definición: Void
2. Los parámetros de dichas funciones van entre paréntesis: ()
3. Los bloques de contenidos se colocan entre llaves: {}
4. Cada línea del código termina en punto y coma : ;

- Ejemplo de ejercicio resuelto:

Escribir un programa que haga parpadear el LED conectado al pin número 13, de manera que esté encendido durante 1 segundo y espere otro segundo antes de volverse a encender; repitiendo este proceso una y otra vez, indefinidamente.

```
void setup()  {
pinMode(13, OUTPUT);
}
void loop() {
        digitalWrite(13, HIGH);
        delay(1000);
        digitalWrite(13, LOW);
        delay(1000);
}
```

:

En este programa hemos utilizado varios comandos:

pinMode (número de pin, INPUT /OUTPUT): se utiliza para configurar

el pin como entrada digital (INPUT) o como salida digital

(OUTPUT).

digitalWrite (número de pin, HIGH /LOW): hacemos que el pin

que numeremos, de una salida de 5 V (HIGH) o 0 V (LOW), es

decir, un 1 o un 0.

delay (tiempo): detiene el programa el tiempo que le indiquemos,

teniendo en cuenta que el tiempo se expresa en milisegundos

(1 000 sería 1 segundo).

Variables

Otro recurso de Arduino son las variables. Una variable es un valor que Arduino almacena para usarlo o modificarlo cuando se necesite. Las variables se declaran normalmente al principio del programa, porque así pueden usarse en cualquier función, aunque no es necesario poner en ese momento un valor concreto, sino que puede hacerse a lo largo del programa (ejemplo: int x y más adelante x=4).

El comando más usado para declarar una variable es int. Se aconseja nombrar las variables con palabras que nos recuerden lo que estamos almacenando en ellas y si usamos más de una palabra, se usa normalmente lo que se llama «notación en joroba de camello» que consiste en poner en mayúscula la siguiente palabra. Por ejemplo:

tiempoEspera lecturaSensor.

Si utilizáramos variables en el ejemplo del programa anterior, sería:

```
int ledPin=13;

int t=1000;

void setup() { pinMode(ledPin,OUTPUT);

}

void loop() {

digitalWrite(ledPin,HIGH);

delay(t);

digitalWrite(ledPin,LOW);

delay(t);

}
```

La diferencia es que nos permitiría modificar los valores del tiempo y del pin en el que hemos situado el led.

5. Comandos condicionales

Cuando programamos con Arduino, habitualmente necesitamos que el código interaccione con las entradas y ejecute una orden u otra según el valor obtenido; esta operación se hace con los comandos condicionales; uno de ellos, que utilizaremos en las prácticas propuestas es:

if (condición) {qué debe realizar el programa}

else {qué hará el programa en caso de no cumplirse la condición}.

Conviene saber que para hacer comparaciones entre dos valores

se utilizan los símbolos, que en Arduino se denominan operadores:

< (menor que), > (mayor que), >= (mayor o igual que), <= (menor o igual que). Si queremos igualar: == (igual que). Se pone el doble signo = para que Arduino entienda que no estamos asignando un valor, sino haciendo una comparación.

6. Verificación y corrección de errores

Una vez escrito el programa, deberemos comprobar su sintaxis haciendo clic en el botón. Si no aparece ningún mensaje en rojo en la parte inferior, significará que es correcto y la compilación del programa ha terminado.

Ya solo queda darle al icono Cargar que volverá a comprobar el programa y lo enviará a través del cable USB a la controladora.

Por tanto, para empezar a trabajar con Arduino tendremos que descargar el programa desde:

https://www.arduino.cc/en/Guide/HomePage

Prácticas que realizaremos:

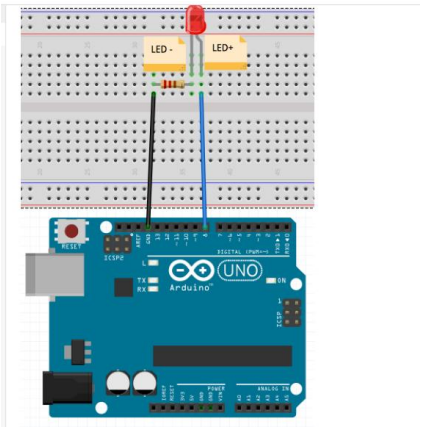
1. Conseguir que parpadee un LED.
2. Controlar un dispositivo con pulsador
3. Construir un semáforo con dos LED de distinto color, rojo y verde con pulsador

Programas ejemplos:

1. Parpadeo de un led:

```
#define pinLED 8

void setup() {

 pinMode(pinLED, OUTPUT);

}

void loop() {

  digitalWrite(pinLED, HIGH); // enciende el LED.

  delay(500); // retardo en milisegundos

  digitalWrite(pinLED, LOW); // apaga el LED.

  delay(500);

}
```

Conexión placa protoboard y placa Arduino para programa anterior

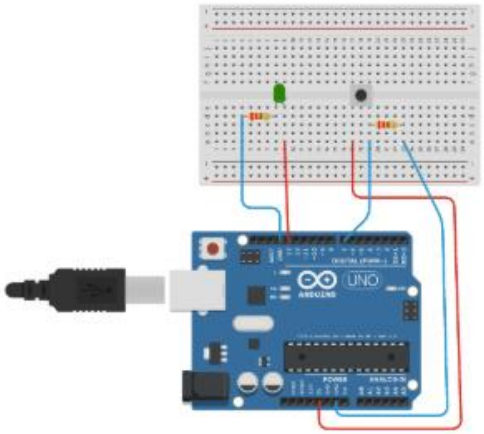


Es importante recordar las características de funcionamiento de un LED. Estos diodos funcionan a 2 o 3 voltios con corrientes de unos 15 a 20 mA, por lo que si la placa da 5 voltios necesitara una resistencia en serie que absorba esos 2 ó 3 voltios de más, bastaría con una resistencia de 220 Ω.

2. Control de un dispositivo a través de un pulsador:

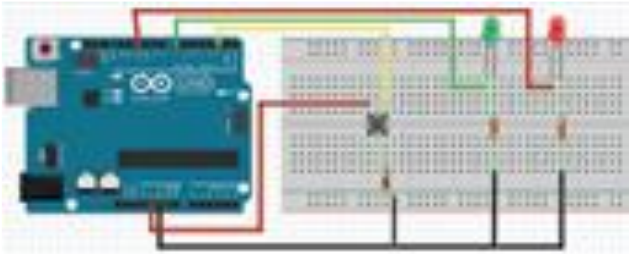
```
const int LED =13;
const int BOTON = 7;
int val = 0; //val se emplea para almacenar el estado del boton
int state = 0; // 0 LED apagado, mientras que 1 encendido
int old_val = 0; // almacena el antiguo valor de val
void setup(){  // definir si la variable es de entrada // o salida.
 pinMode(LED,OUTPUT); // establecer que el pin digital es una señal de salida
 pinMode(BOTON,INPUT); // y BOTON como señal de entrada
}
void loop() {      // loop = realice un lazo continuamente
val= digitalRead(BOTON); // lee el estado del Boton
if ((val == HIGH) && (old_val == LOW)){
state=1-state;
delay(10);
}
old_val = val; // valor del antiguo estado
if (state==1){
 digitalWrite(LED, HIGH); // enciende el LED
}
else{
 digitalWrite(LED,LOW); // apagar el LED
}
}
```

Conexión placa protoboard y placa Arduino para programa anterior:



3. Construir un semáforo con dos LED de distinto color, rojo y verde con pulsador

Montaje



Solución:

```
// declarando parametros
int led_verde = 13;
int led_rojo = 8;
const int BOTON = 7; // pin de entrada botón
int val = 0; //val se emplea para almacenar el estado
// del botón
int state = 0; // 0 LED apagado, mientras que 1 encendido
int old_val = 0; // almacena el antiguo valor de val// setup de parámetros
void setup() {
// se indica que cada pin es de salidad OUTPUT.
pinMode(led_verde, OUTPUT);
pinMode(led_rojo, OUTPUT);
}
// lazo a ejecutar continuamente una vez cargado el código en el arduino
void loop(){
val= digitalRead(BOTON); // lee el estado del Boton
// chequear si el boton esta presionado o no
if ((val == HIGH) && (old_val == LOW)) {
state=1-state;
delay(10);
}
old_val = val; // val is now old, let's store it
if (state==1) {
```

```
digitalWrite(led_verde,HIGH); // encender LED verde
delay(5000); // mantener por 5 segundos
digitalWrite(led_verde,LOW); // apagar LED verde
digitalWrite(led_rojo,HIGH); // encender LED rojo
delay(4000);
digitalWrite(led_rojo,LOW); // encender LED rojo
}
}
```

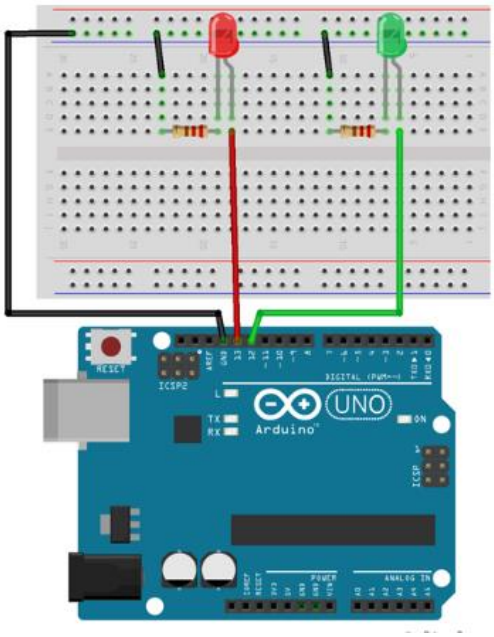
Teniendo en cuenta los dos programas anteriores realiza los montajes siguientes:

1. Conseguir que parpadeen dos LED, primero de forma simultánea y después de forma alterna.
2. Construir un semáforo con tres LED, rojo, verde y amarillo.
3. Control de un semáforo con pulsador.

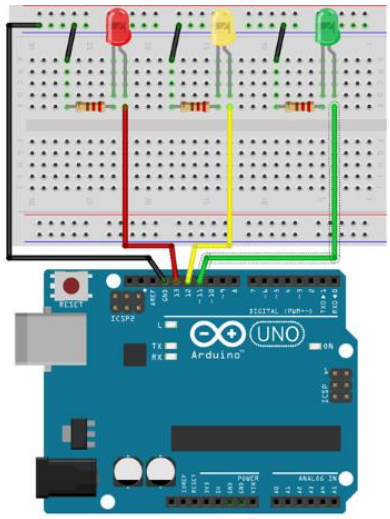
SOLUCIÓN ACTIVIDAD 1:

Luz alterna

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);
  digitalWrite(12, LOW);
  delay(1000);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  delay(1000);
}
```



SOLUCIÓN ACTIVIDAD 2:



*** Semáforo simple**

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
  digitalWrite(13, LOW);
  digitalWrite(12, LOW);
  digitalWrite(11, LOW);
}
void loop() {
  digitalWrite(13, LOW);
  digitalWrite(11, HIGH);
  delay(5000);
  digitalWrite(11, LOW);
  digitalWrite(12, HIGH);
  delay(1000);
  digitalWrite(12, LOW);
  digitalWrite(13, HIGH);
  delay(5000);
}
```

SOLUCIÓN EJERCICIO 3: REALIZAR SECUENCIA BASÁNDOTE EN EL EJEMPLO 3 RESUELTO.

En el siguiente enlace se ve el resultado. https://www.youtube.com/watch?v=zUxhKDf2eD4