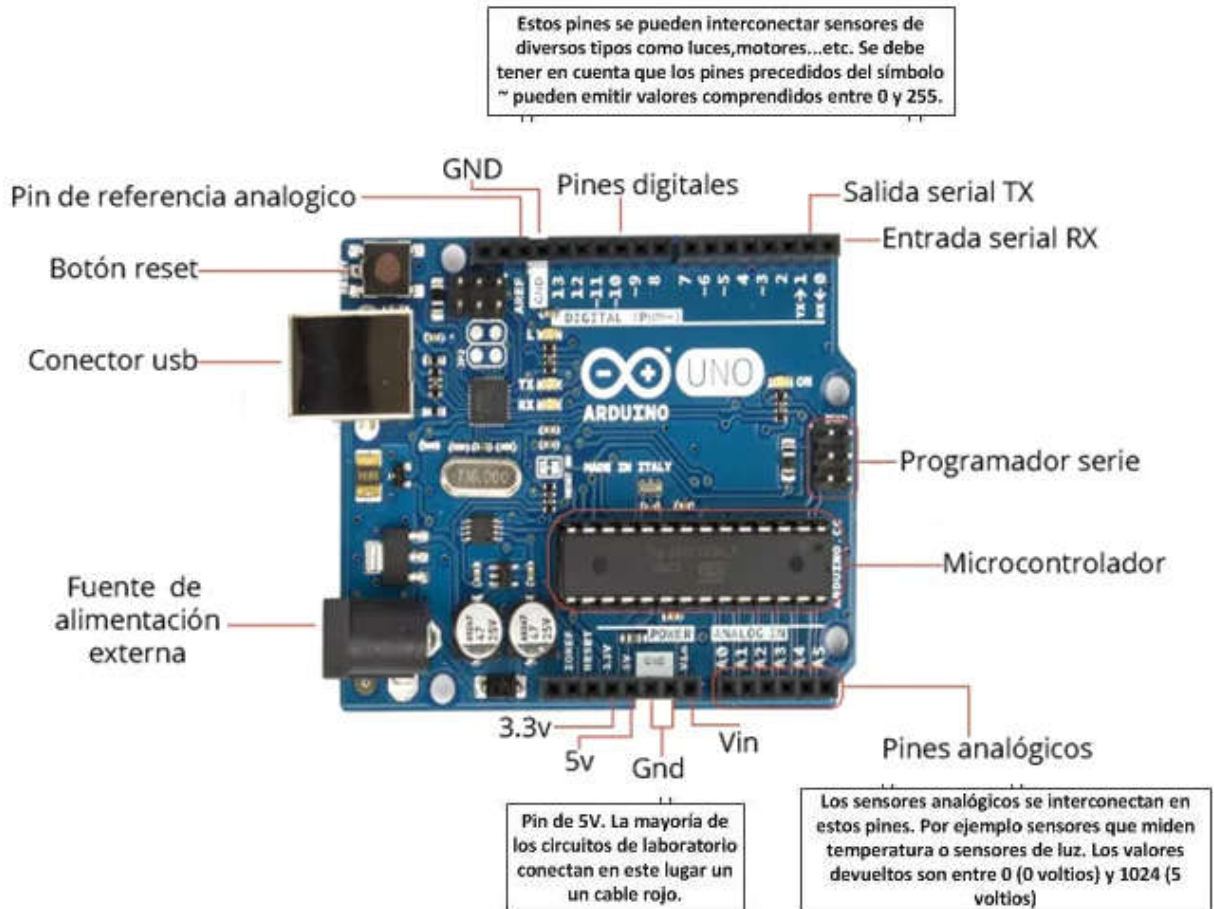




Tecnologías para la Automatización – Práctica Parte 2

Arduino:

La estructura básica de los ejercicios está diseñada con el objetivo de poder comprender entre otros el lenguaje de programación de Arduino y la instalación de componentes electrónicos.



Objetivos:

- Reconocer partes de la Placa Arduino (Uno).
- Aprender a conectar los Pines de la Placa Arduino.
- Familiarizarse con el Entorno de Programación.
- Reconocer las partes de un programa en Arduino IDE.
- Conocer Órdenes Básica de Programación.



1. LED Parpadeante

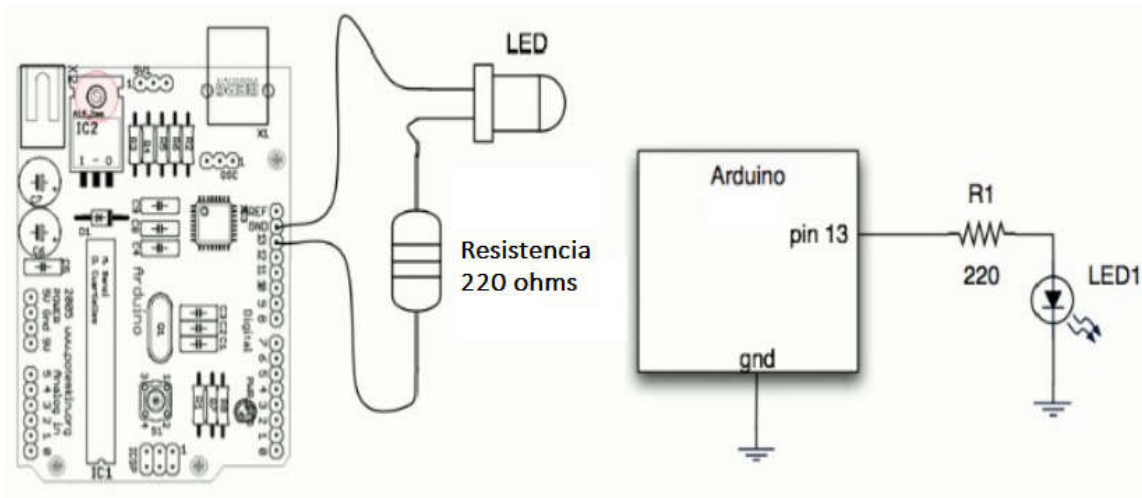
Recordemos que en el **“Void Setup()”** se coloca la configuración y **“Void Loop()”** se coloca el código de las acciones.

Código:

Void Setup() Declaramos al Pin 13 como Salida.

“Void Loop()” Mandamos señal Alta al pin 13 (Encendemos), Una Espera de 1 minuto (1000 milisegundo), mandamos señal Baja al pin 13 (apagamos), otra Espera de 1 minuto (1000 milisegundo) y se repetirá el Ciclo.

Conexiones:



2. Secuencia de LEDs Parpadeantes

Código:

Void Setup() Declaramos a los Pines 5, 6, 7 y 8 como Salida.

Void Loop() Mandamos señal Alta al pin 5 (Encendemos), Una Espera de Medio minuto (500 milisegundo), mandamos señal Baja al pin 5 (apagamos), otra Espera de medio minuto (500 milisegundo), repetimos la misma

Acción para los pines 6, 7 y 8 y siempre se repetirá el Ciclo.

Este se ejercicio se realizará a través de varias formas de Códigos:

Código básico:

Se trabajará directamente como en el ejercicio 1, trabajando sobre los pines de la tarjeta Arduino.

Código con Uso de Variables:

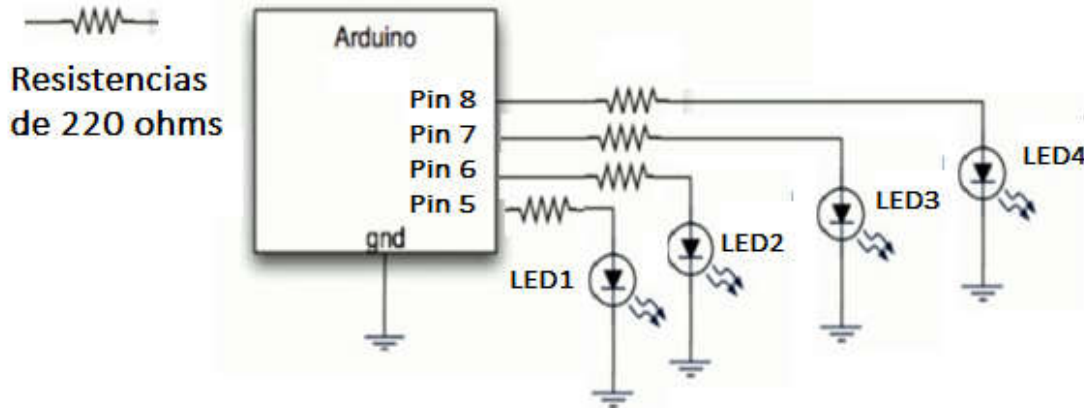
Se verá cómo se realiza la declaración de las variables y su aplicación en el código.

Código con Uso de Función y Variables:



Se verá cómo se realiza la declaración de las variables y de las funciones y como se aplica en el código.

Conexiones:



Recordemos que una variable es un espacio de memoria que se reserva con un nombre para alojar una información temporalmente.

Una función es una secuencia de instrucciones agrupadas bajo un nombre y que se ejecutarán al momento de ser llamada en la ejecución de "Void Loop()".

Un vector es un arreglo unidimensional, es decir un conjunto de variables que llevan el mismo nombre y poseen una determinada cantidad de filas o ubicaciones.

3. Secuencia de LEDs Varios colores:

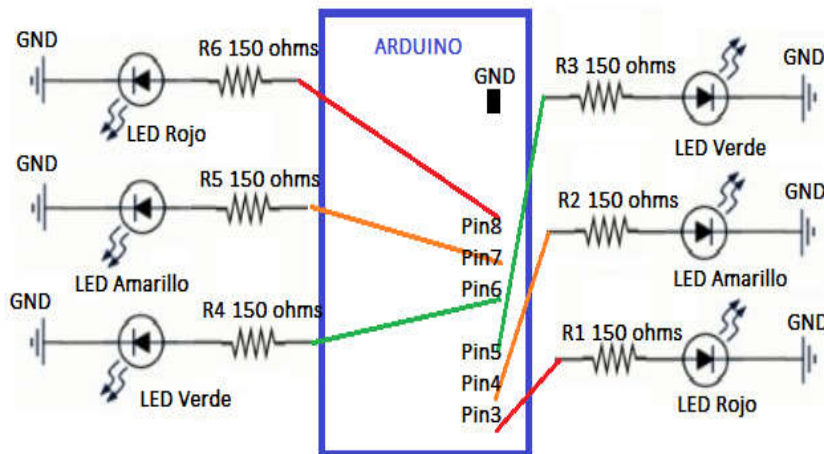
Se trata de realizar una secuencia de luces encendiendo dos Led del mismo color por vez.

Código:

Void Setup() Declaramos a los Pines 3, 4, 5, 6, 7 y 8 como Salida.

Void Loop() Mandamos señal Alta a los pines 4 y 7 (que tiene los dos el mismo color de Leds y los encendemos), Una Espera de Medio minuto (500 milisegundo), mandamos señal Baja a los pines 4 y 7 (y los apagamos), otra Espera de medio minuto (500 milisegundo), repetimos la misma acción para los pines 3 y 8 y al final para los Pines 5 y 6 y siempre se repetirá el Ciclo.

Conexiones:



4. SOS con Zumbador:

Se trata de realizar señales sonoras y lumínicas pidiendo SOS en código Morse.

Código:

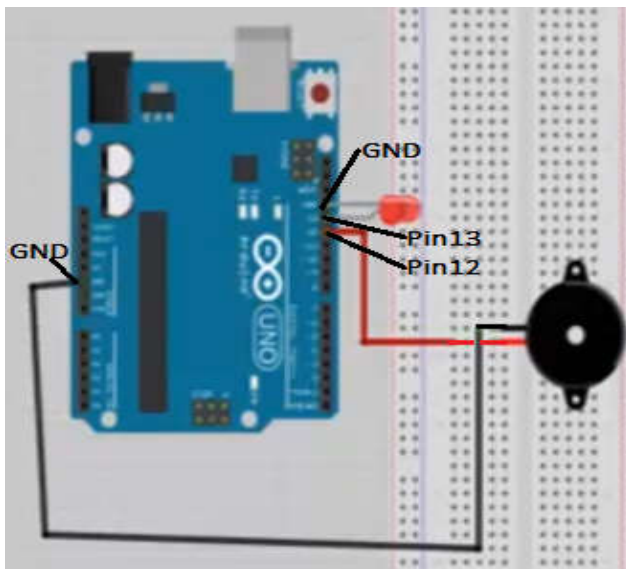
Primero declararemos las variables tanto para el pin del Led como para el Zumbador.

Void Setup() Declaramos a los Pines del Led y de Zumbador como Salida.

Void Loop() Llamamos a las funciones S, O y S con una espera entre cada una de ellas.

Debajo y fuera del **Void Loop** creamos las funciones S, O y S.

Conexiones:





5. Encender LED con Pulsador:

Se trata de encender un LED cuando pulsamos el Pulsador.

Código:

Primero declararemos las variables tanto para el pin del Led como para el Pulsador y otra que nos ayudará a controlar el estado en el que estamos.

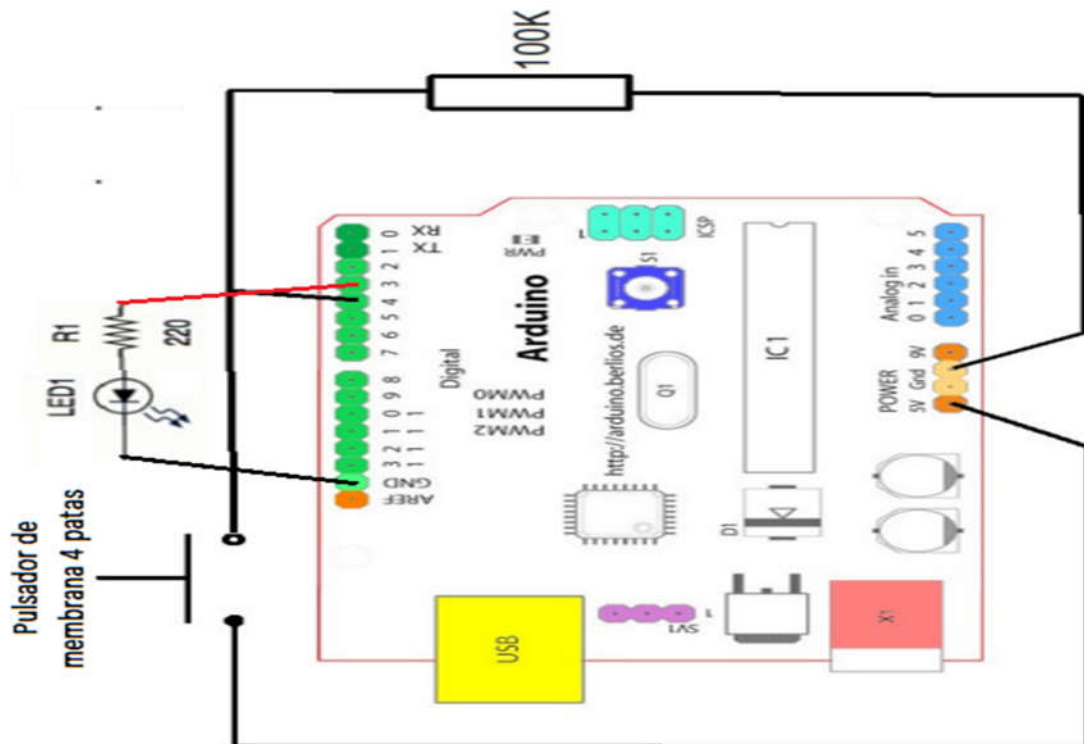
Void Setup() Declaramos a los Pines del Pulsador como Entrada y del LED como Salida e iniciaremos al LED apagado.

Void Loop() Con un Bucle preguntamos el estado del Pulsador y la variable estado recogerá el estado del LED.

A continuación, el Pin del LED tomará el estado contrario al que se encuentra el LED.

Colocamos otro bucle que controle los rebotes del Pulsador que mientras mande señales de rebotes no haga nada.

Conexiones:



6. Encender LED RGB:

Se trata de encender cada uno de los colores de un LED RGB.

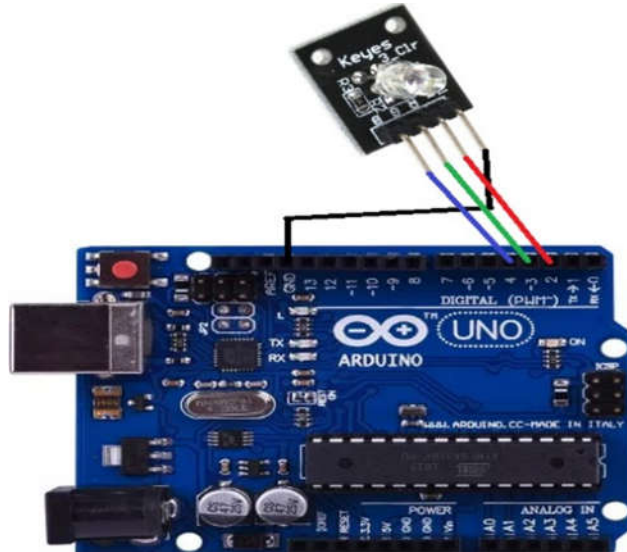
Código:

Void Setup() Configuramos como salidas los Pines que le asignaremos a cada uno de los Pines del LED RGB.



Void Loop() Haremos que se encienda uno de los colores (rojo) mientras los otros dos permanecen apagados, le damos una espera de un minuto y repetimos la acción para cada uno de los colores siempre con una espera de un minuto entre el rojo, el verde y el azul.

Conexiones:



7. Sensor PIR (de movimientos) Encender LED por movimiento

Se trata de encender un LED cada vez que el sensor PIR detecte un movimiento.

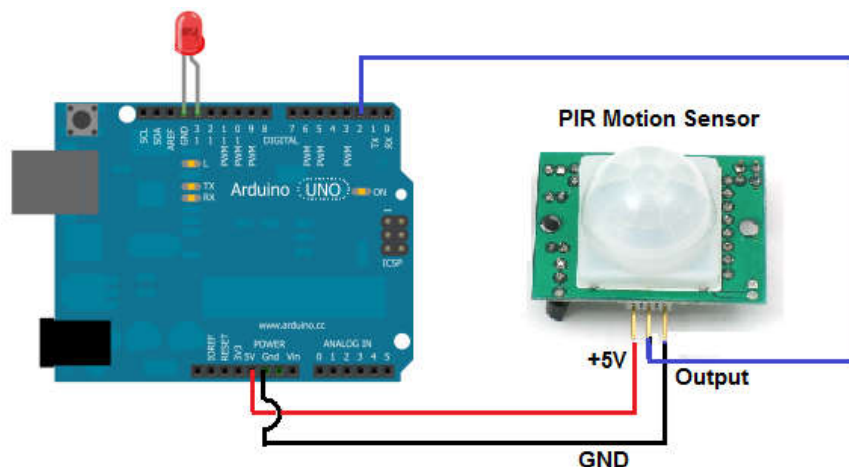
Código:

Declararemos las variables tanto para el Sensor como para el led asignándoles un Pin para cada uno de ellos

Void Setup() Configuramos como salidas el Pin del LED y como entrada al Pin del Sensor y abrimos nuestro puerto serial.

Void Loop() Leeremos el Pin del sensor y si detecta movimiento mandará un mensaje por serial y encenderá el LED y después de una espera lo apagará.

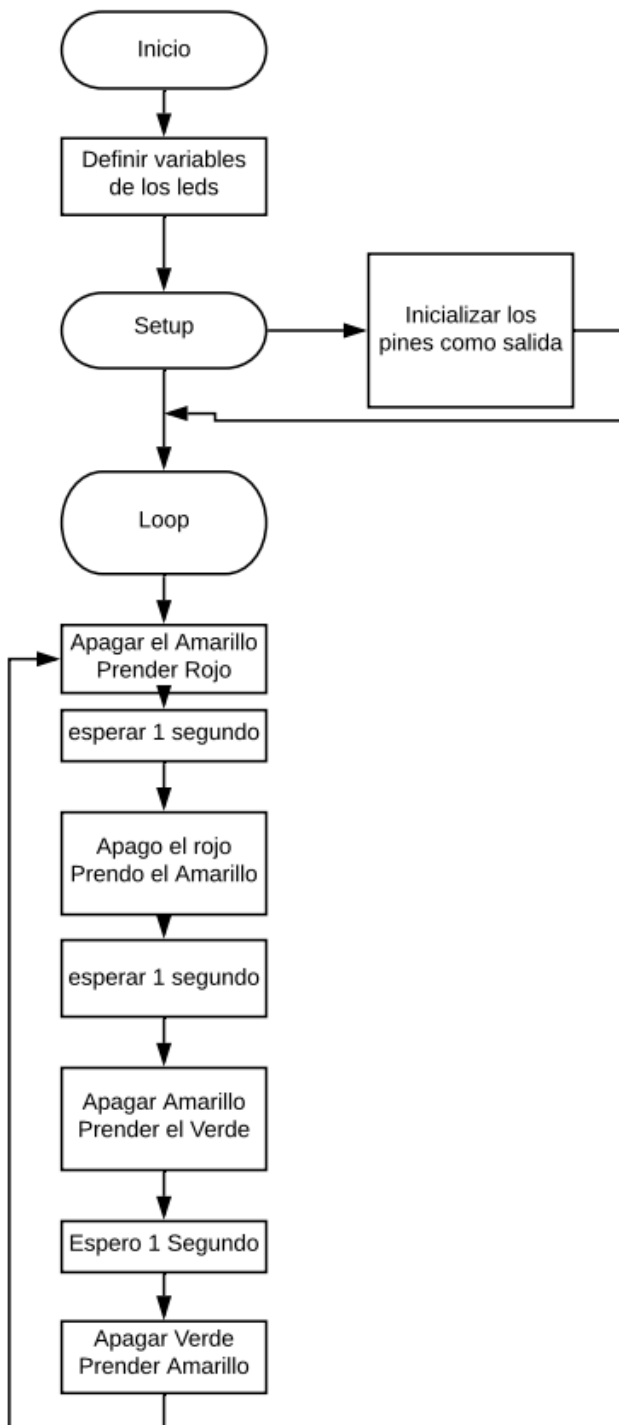
Conexiones:





8. Semáforo

Codificar el siguiente diagrama de flujo en Arduino.

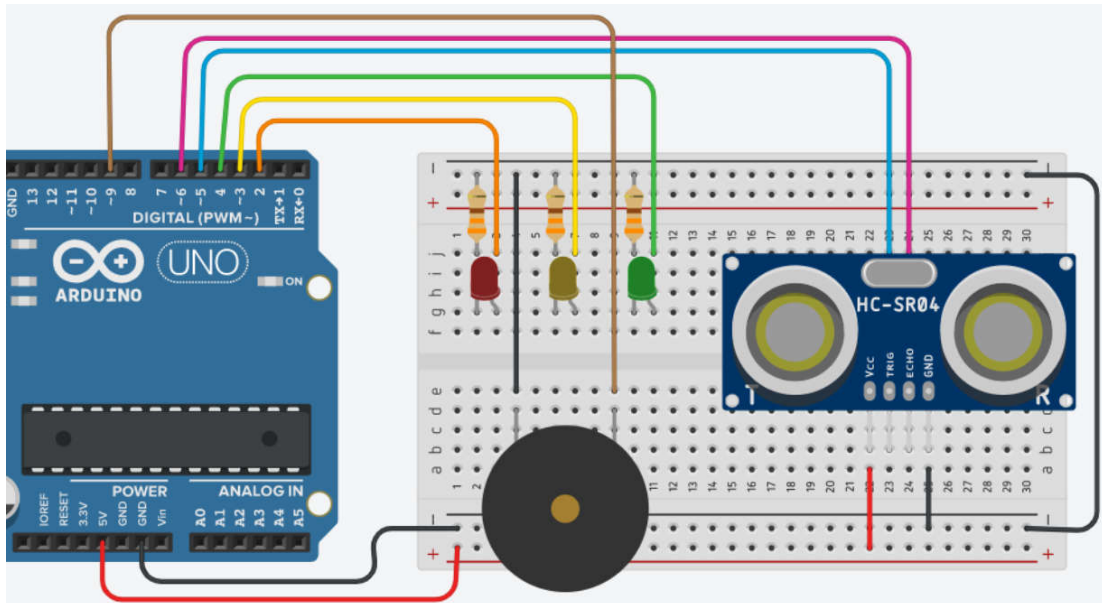
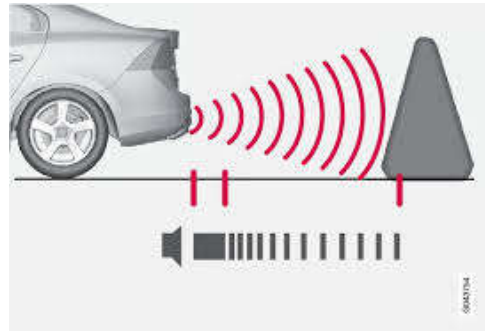




9. Indicador de estacionamiento para autos

Crear un prototipo que permita avisar al chofer la distancia de objetos en el proceso de aparcarse un auto. Para ello usaremos:

- 1 Arduino uno
- 1 Sensor ultrasónico
- 3 Resistencias 300Ω
- 3 Leds (rojo, amarillo, verde)
- 1 Buzzer
- 1 Protboard





10. Motor CC controlado mediante el integrado L293D.

Desarrolla un sistema de control utilizando una placa Arduino para controlar un motor CC controlado mediante el integrado L293D, utilizar 2 LEDs de color rojo y verde. El motor debe girar en las dos direcciones (adelante y atrás), cuando va hacia adelante deben parpadear ambos LEDs y cuando va hacia atrás debe parpadear el LED rojo.

Requisitos:

1. Utilizar una placa Arduino con al menos cuatro pines disponibles para el control.
2. Conectar el motor y los LEDs a los pines de salida correspondientes.
3. Implementar las funciones necesarias para controlar el sistema:
 - **avanzar()**: Activar el motor para girar en una dirección específica.
 - **retroceder()**: Activar el motor para girar en la dirección opuesta.
 - **parar()**: Detener el motor y apagar los LEDs.
 - **parpadear()**: Hacer parpadear los LEDs en un patrón específico.

En el bucle principal (loop()), realizar las siguientes acciones de forma secuencial:

Girar el motor hacia adelante.

Hacer parpadear los LEDs.

Detener el motor.

Esperar un breve periodo de tiempo.

Cambiar la dirección del motor hacia atrás.

Detener el motor.

Esperar un breve periodo de tiempo.

11. Motor CC controlado mediante el integrado L293D.


Desarrolla un sistema de control utilizando una placa Arduino para controlar un motor CC controlado mediante el integrado L293D, utilizar 2 LEDs de color rojo y verde. Modificar el ejercicio 1 agregando 1 rueda y que pueda girar a la derecha y a la izquierda. Cuando gire a la derecha debe encender la luz roja y cuando doble a la izquierda el led azul.

El programa debe mantenerse girando en círculos 5 segundos a la derecha y luego 5 segundos a la izquierda.

12. control sobre la velocidad de los motores

Aplicar control sobre la velocidad de los motores del ejercicio 11 para que pueda doblar hacia la derecha y a la izquierda suavemente sin girar sobre si mismo. Para esto agregar pines de control de tipo PWM para ambos motores.






ARDUINO

Arduino Cheat Sheet / Acordeón Arduino

Idea adaptada del original Arduino Cheat Sheet por Gavin para "Robots and Dinosaurs".
Referencia de Lenguaje Arduino: <http://arduino.cc/en/Reference/HomePage>



Genuino

Estructura y Flujo

Sintaxis

```
// (Comentario en una línea)
/* (Comentario de múltiple línea) */
#define(x)
#include <NombreLibreria.h>
Estructura básica del programa
void setup() {
  //Corre una tan sola vez
}
void loop() {
  // Se ejecuta repetidamente
}
Estructuras de control
if (x < 5) { ... } else { ... }
while (x < 5) { ... }
do { ... } while (x < 5);
for (int i = 0; i < 10; i++) {
  // ...
}
break; //sale del bucle inmediatamente
continue; //va a la siguiente iteración
switch (miVariable) {
  case 1:
    // ...
    break;
  case 2:
    // ...
    break;
  default:
    // ...
}
return x; // o "return;" para vacíos
```

Variables, Datos y Vectores

Conversiones

```
char()
byte()
int()
float()
long()
```

Calificadores

```
static //persiste entre llamadas
volatile //usa la RAM
const //sólo lectura
PROGMEM //usar la flash
```

Punteros

```
& (referencia: obtener puntero)
* (valor: seguir puntero)
```

Constantes

```
HIGH | LOW
INPUT | OUTPUT
true | false
//Octal (comenzando en 0)
0b10111111 //Binario
0x7B //Hex (hexadecimal)
7U //forzar unsigned
10L //forzar long
10.0 //forzar floating point
2.4e5 //240000
```

Tipos de datos

```
void
boolean (0, 1, true, false)
char (e.j. 'a' -128 a 127)
int (-32768 a 32767)
long (-2147483648 a 2147483647)
unsigned char (0 a 255)
byte (0 a 255)
unsigned int (0 a 65535)
word (0 a 65535)
unsigned long (0 a 4294967295)
float (-3.4028e+38 a 3.4028e+38)
double (igual que los flotantes)
```

Cadenas

```
char S1[8] = "A","r","d","u","i","n","o";
//cadena sin terminación
char S2[8] = "A","r","d","u","i","n","o","\0";
//puede producir error
char S3[] = "terminación nula \0";
//incluye terminación nula
char S4[8] = "arduino";
```

Vectores y matrices

```
int myInts[6]; //vector de 6 enteros
int myPins[] = {2, 4, 8, 3, 6};
int mySensVals[6] = {2, 4, -8, 3, 2};
myInts[0] = 42; //asigna al primero
//en el índice
myInts[6] = 12; //ERROR! El índice va de 0 a 5
//de 0 a 5
```

Funciones Incluidas

I/O Digital

```
pinMode(pin, INPUT, OUTPUT)
digitalWrite(pin, valor)
int digitalWrite(pin)
//Escribe HIGH en entradas para //usar los pull-ups
```

I/O Analógicas

```
analogReference(DEFAULT, INTERNAL, EXTERNAL)
int analogRead(pin)
analogWrite(pin, valor) //PWM
```

Advanced I/O

```
tone(pin, freqhz)
noTone(pin)
shiftOut(pindatos, pinReloj, MSBFIRST, LSBFIRST, valor)
unsigned long pulseIn(pin, HIGH, LOW)
//8 - 180
```

Tiempo

```
unsigned long millis() //desbordamiento en 50 días
unsigned long micros() //desbordamiento en 78 minutos
delay(ms)
delayMicroseconds(us)
```

Matemáticas

```
min(x, y)
max(x, y)
abs(x)
sin(rad)
cos(rad)
tan(rad)
sqrt(x)
pow(base, exponente)
constrain(x, valMin, valMax)
map(val, deAtoB, deATo, deBTo, aATo)
```

Números aleatorios

```
randomSeed(semilla) //long ó int
long random(max)
long random(min, max)
```

Bits y Bytes

```
lowByte(x)
highByte(x)
bitRead(x, bitn)
bitWrite(x, bitn, bit)
bitSet(x, bitn)
bitClear(x, bitn)
bit(bitn) // bitn: 0=LSB 7=MSB
```

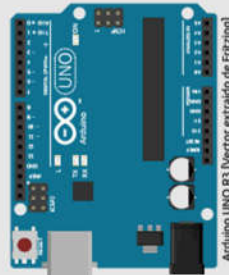
Interrupciones Externas

```
attachInterrupt(interrupt, func, LOW, CHANGE, RISING, FALLING)
detachInterrupt(interrupt)
interrupts()
noInterrupts()
```

Bibliotecas

Serie

```
begin(1200, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200)
//Puede ser cualquier número
end()
byte read()
byte peek()
flush()
print(miDatos)
println(miDatos)
flush()
EEPROM() #include <EEPROM.h>
byte read(dirInterna)
write(dirInterna, miByte)
Servo #include <Servo.h>
attach(pin, min_us, max_us)
write(angulo) // 0, 180
writeMicroseconds(us)
//1000-2000; 1500 es en medio
read() //0 - 180
attached() //regresa booleano
detach()
SoftwareSerial(RxPin, TxPin)
begin(long velocidad) //hasta 9600
char read() //espera los datos
print(miDatos)
println(miDatos)
Wire #include <Wire.h> //para PC
begin() //se une a maestro
requestFrom(dirección, cuenta)
beginTransmission(dir) // Paso 1
send(miByte) // Paso 2
send(char * miCadena)
endTransmission() // Paso 3
byte available() // Num de bytes
byte receive() //Regresa el sig byte
onReceive(func) //Regresa el sig byte
onRequest(func)
```



Arduino Uno R3 (Vector extraído de Fritzing)

Este trabajo está bajo licencia Atribución-Compartir Igual 3.0

- Adaptación por Liffon
- Versión SVG por Frédéric Dufour
- Traducción al español de Antonio Maldonado
- Diseño y adaptación por Karla L. Hdz
- Inspirado en adaptación de SparkFun Electronics
- Paleta de colores tomada del "Arduino Day"

MAS INFORMACION EN: