## Estructura del Programa

```
void setup()
{
  // Se ejecuta una sola vez cuando
  // el  Arduino prende o se reinicia
}
void loop()
{
  // Se ejectua repetidamente todo
  // el codigo dentro de loop()
}
```

## Estructuras de Control

```
if ( Condición )
 { // ejecuta si cumple condición }
else
 { // ejecuta si no cumple condición}

while ( Condición )
 {  /* ejecuta repetidamente
    mientras cumpla la condición */  }

for(valor Inicial;Condición;Incremento)
 {//ejecuta mientras cumpla condición}

switch ( miVariable )
 {
    case valor1:
    //ejecuta si valor1==miVariable
    break;
    case valorN:
    //ejecuta si valorN==miVariable
    break;
    .....
    Default:
    //ejecuta si no cumple ninguno
  }
```

## Variables

| | |
|---|---|
| void | vacio |
| boolean | true, false |
| char | caracter [-128 a 127] |
| byte | entero [0 a 255] |
| word | entero [0 a 65535] |
| int | entero [-32768 a 32767] |
| long | entero [-2147483648 a 2147483647] |
| float | decimal [-3.4028e+38 a 3.4028e+38] |
| double | decimal [-3.4028e+38 a 3.4028e+38] |

## Comentarios

```
// Comentario de una linea

/*  Este es un comentario de
    varias lineas  */
```

## Operadores

### Operadores Aritmeticos

= operador de asignacion
+ Suma  - resta
* Multiplicacion
/ División  % Modulo

### Operadores Comparación

== igual a  != diferente a
< menor que  > mayor que
<= menor o igual que
>= mayor o igual que

### Operadores Booelanos

&& and  || or  ! not

### Operadores Compuestos

++ incremento
-- decremento
+= suma compuesta
-= resta compuesta
*= multiplicacion compuesta
/= division compuesta
&= AND binario compuesto
|= OR binario compuesto

### Operadores a nivel de bit

& AND binario  | OR binario
^ XOR binario  ~ NOT
<< desplazamiento a la izq.
>> desplazamiento a la der.

## Constantes

HIGH | LOW
INPUT | OUTPUT
true | false

```
123          //Decimal
0123         //Octal
0b11001111   //Binario
0xF3         //Hexadecimal
2.4e5        //240000
```

## Funciones del Arduino

### Entradas y Salidas Digitales

```
pinMode(pin,INPUT/OUTPUT);
//Configura como entrada o salida un pin (0-13) del Arduino
//INPUT ← Entrada   OUTPUT → Salida
digitalWrite(pin,LOW/HIGH);
//Escribe en el pin del arduino el estado logico LOW o HIGH
//LOW → Bajo(0)   HIGH → Alto(1)  | Solo en los pines de salida
digitalRead(pin);
//Lee el estado logico del pin del Arduino ( LOW / HIGH )
```

### Entradas y Salidas Analógicas

```
analogRead(pin);
//Lee el valor (0-1023) analógico del pin (A0-A5) del Arduino
analogWrite(pin, valor);
//Escribe en el pin (3,5,6,9,10,11) un valor (0-255), Salida PWM
```

### Tiempo

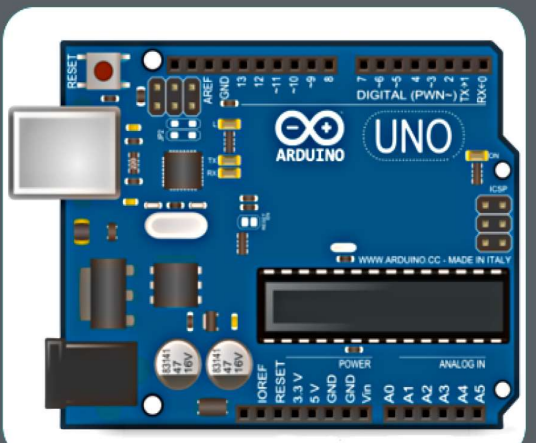
```
delay(tiempo);
//Genera un retardo, el tiempo está en milisegundos 1s=1000ms
millis();
/*Devuelve el tiempo de funcionamiento del Arduino en
milisegundos*/
micros();
/*Devuelve el tiempo de funcionamiento del Arduino en
microsegundo*/
```

### Salidas Avanzadas

```
tone(pin,frecuencia);
//Genera un tono en el pin y la frecuencia establecida
tone(pin,frecuencia,tiempo)
//Genera un tono en el pin y la frecuencia durante un tiempo(ms)
noTone(pin);
//Detiene el tono generado por la función tone()
```

### Matemáticas

```
min(x,y)    //Calcula el valor menor entre "x" y "y"
max(x,y)    //Calcula el valor mayor entre "x" y "y"
abs(x)      //Calcula el valor absoluto de "x"
sin(rad)    //Calcula el valor seno en radianes
cos(rad)    //Calcula el valor coseno en radianes
tan(rad)    //Calcula el valor tangente en radianes
sqrt(x)     //Calcula la raiz cuadrada de "x"
pow(base,exponente)
//Calcula el valor de un numero elevado a la potencia
constrain(x, valMin, valMax)
//limita a “x” entre el “valMin” y “valMax”
map(x, rango1Min, rango1Max, rango2Min, rango2Max)
//Modifica el valor de “x” del rango1 proporcionalemte al rango2
```




## Librerias

### Comunicación Serial

```
Serial.begin(baudio)
//Abre el puerto serial y establece la velocidad
//de comunicacion en baudios
Serial.print(dato)
//Imprime el dato en el puerto serie (envia)
Serial.println(dato)
//Imprime el dato seguido de retorno de carro
Serial.Read()
//Lee un byte (caracter) desde el puerto serie
Serial.available()
//Devuelve un entero con el numero de byte
//(caracteres) disponibles para leer en el buffer
```

### Memoria EEPROM

```
#include<EEPROM.h>
//cargamos la libreria EEPROM al programa
EEPROM.read(direccion)
//Lee el Byte guardado en “direccion”
EEPROM,write(direccion,dato)
//Guarda el “dato” en la “direccion”
```

# Prototipado con ARDUINO