ESTRUCTURA DEL PROGRAMA

void setup() {

ordenes

}

función solo se ejecutará una vez, después de cada encendido o reinicio de la placa Arduino

void loop() {

ordenes

}

Se repite continuamente

TIPOS DE DATOS

int : almacena números enteros en un rango -32,768 a 32,767

long: almacena números enteros en un rango -2,147,483,648 to 2,147,483,647.

bool: A booltiene uno de dos valores, true o false

float: contiene números decimales

ORDENES DE ENTRADA Y SALIDA

pinMode(pin, modo)

pin: el número de pin que queremos activar como entrada o salida.

modo: INPUT, OUTPUT, or INPUT_PULLUP

digitalWrite(pin, valor)

pin: el número de pin

valor: HIGH or LOW

digitalRead(pin)

pin: el número de pin digital que quieres leer (tipo int)

analogRead(pin)

pin: el número de pin analógico del que se quiere leer la entrada

analogWrite(pin, valor)

pin: el número de pin en el que queremos escribir.

valor: la duración del pulso: entre 0 y 255

ESTRUCTURA DE CONTROL

if (condition) {

ordenes

}

Ejemplo

if (x > 120) {

digitalWrite(LEDpin1, HIGH);

digitalWrite(LEDpin2, HIGH);

}

Operadores de comparación:

x == y (x es igual a y)

x != y (x no es igual a y)

x < y (x es menor que y)

x > y (x es mayor que y)

x <= y (x es menor o igual a y)

x >= y (x es mayor o igual a y)

for (initialization; condition; increment) {

ordenes

}

Parámetros

initialization: ocurre primero y exactamente una vez.

condition: cada vez que se pasa por el bucle, conditionse prueba; si es trueel bloque de declaración y se ejecuta el incremento , entonces la condición se prueba nuevamente. Cuando la condición se vuelve false, el ciclo termina.

increment: ejecutado cada vez que pasa por el bucle cuando conditiones true.

Ejemplo

```
for (int x = 2; x < 100; x = x * 1.5) {
  println(x);
}
```

Genera: 2,3,4,6,9,13,19,28,42,63,94

OPERADORES COMPUESTOS

++

x++; // incrementa x en uno y devuelve el Viejo valor de x

++x; // incrementa x en uno y devuelve el nuevo valor de x

--

Funciona igual que ++ pero ahora decrementa

OPERADORES BOOLEANOS

!

El NOT lógico da como resultado a true si el operando es falsey viceversa

&&

El resultado lógico AND es true solo si ambos operandos son true.

Ejemplo

```
if (digitalRead(2) == HIGH && digitalRead(3) == HIGH) { // da
verdadero si la entrada 2 y 3 están en 1 //
ordenes
}
```

||

El OR lógico da como resultado a true si cualquiera de los dos operandos es true.

Ejemplo

```
if (x > 0 || y > 0) { // if  x or y son mayores que cero da
verdadero  //
statements
}
```

OTRAS

delay(ms)

ms: la duración de la pausa en número de milisegundos