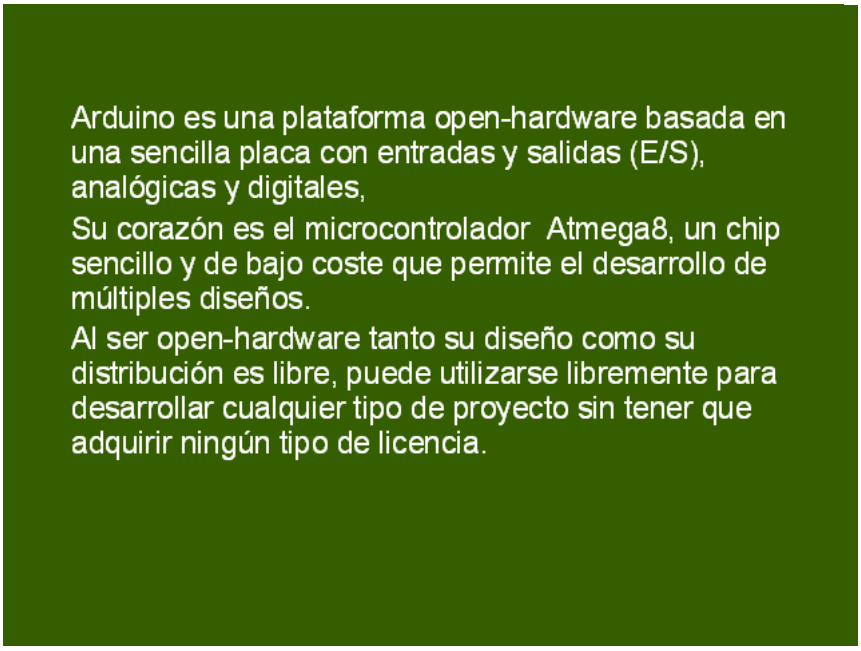
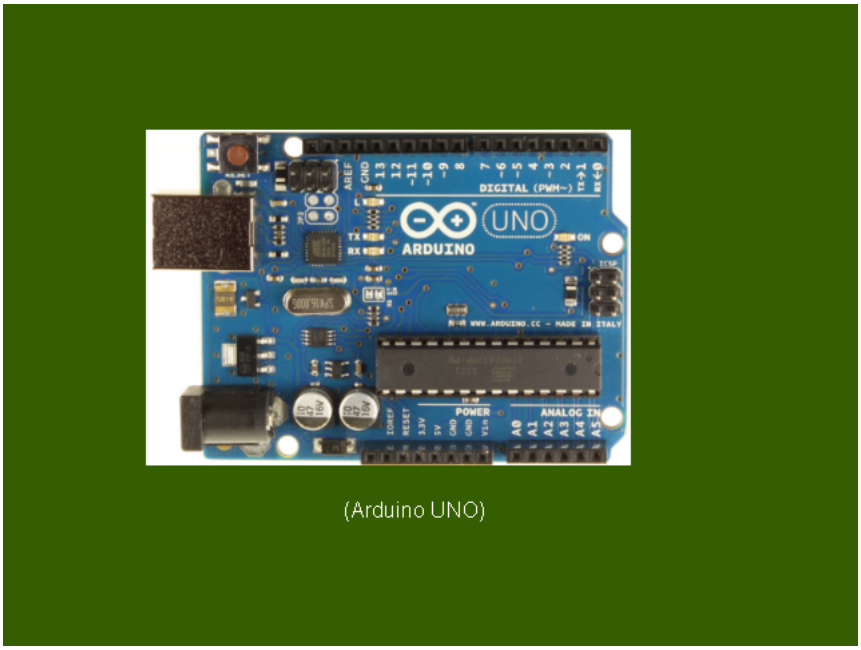
Introducción a Arduino

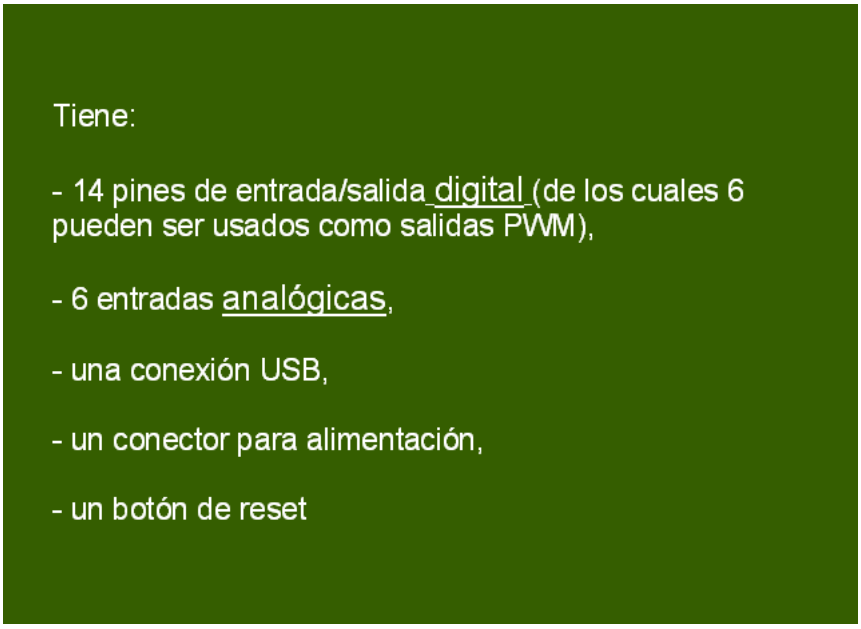
Enviado por Pablo Turmero



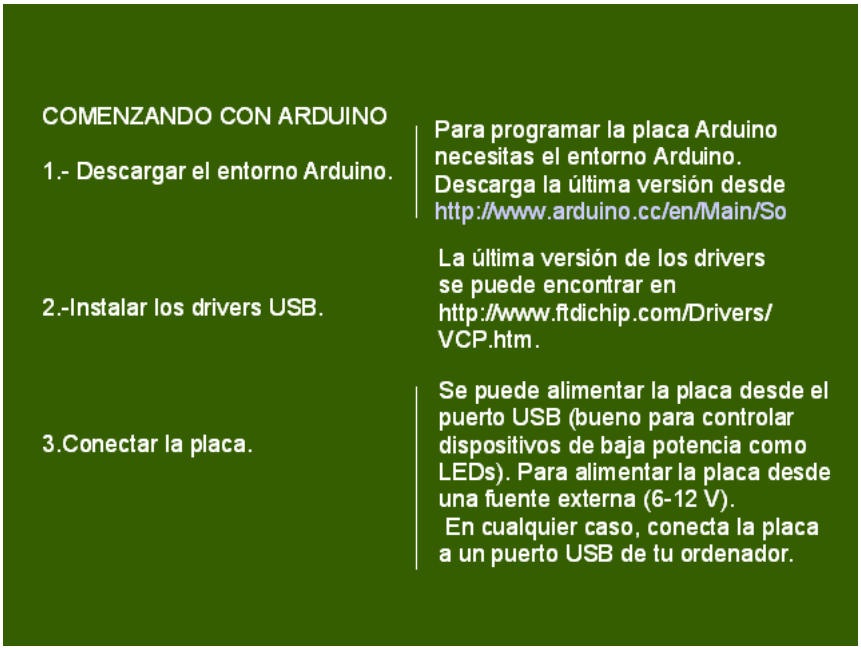
Arduino es una plataforma open-hardware basada en una sencilla placa con entradas y salidas (E/S), analógicas y digitales, Su corazón es el microcontrolador Atmega8, un chip sencillo y de bajo coste que permite el desarrollo de múltiples diseños. Al ser open-hardware tanto su diseño su distribución es libre, puede utilizarse libremente para desarrollar cualquier tipo de proyecto sin tener que adquirir ningún tipo de licencia.



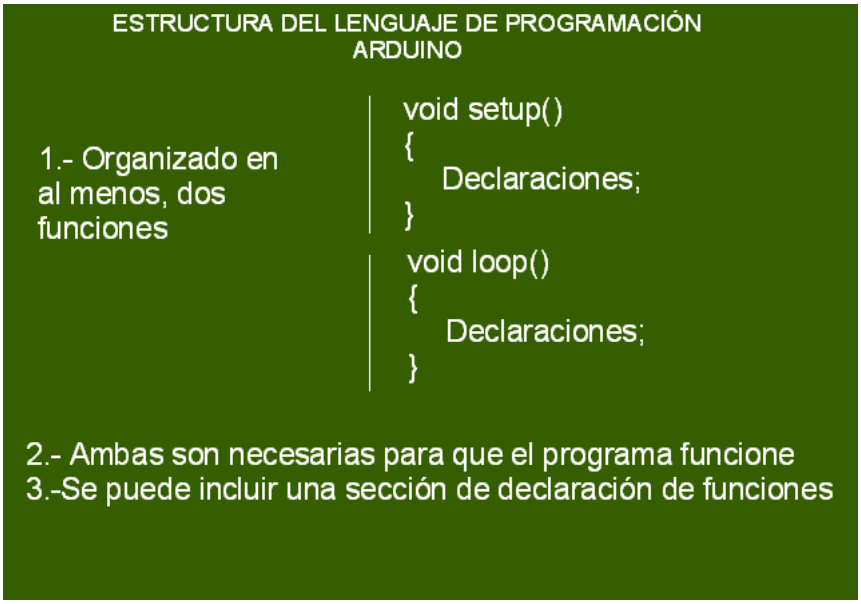
(Arduino UNO)



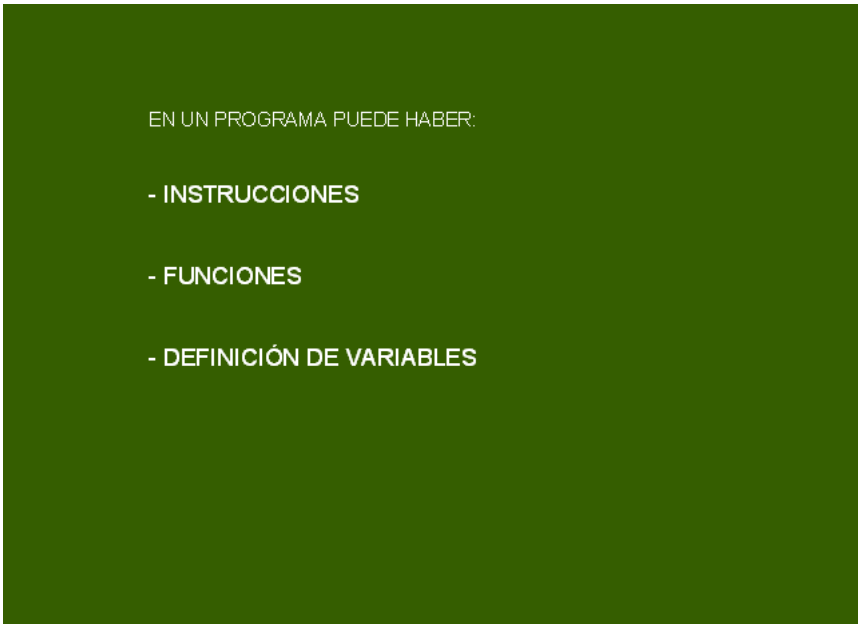
Tiene: - 14 pines de entrada/salida digital (de los cuales 6 pueden ser usados como salidas PWM), - 6 entradas analógicas, - una conexión USB, - u conector para alimentación, - un botón de reset



COMENZANDO CON ARDUINO 1.- Descargar el entorno Arduino. 2.-Instalar los drivers USB. 3.Conectar la placa. Para programar la placa Ardu necesitas el entorno Arduino. Descarga la última versión desde http://www.arduino.cc/en/Main/Software La última versión de los drivers se pue encontrar en http://www.ftdichip.com/Drivers/ VCP.htm. Se puede alimentar la placa desde el puerto USB (bueno para controlar dispositivos de potencia como LEDs). Para alimentar la placa desde una fuente externa (6-12 V). En cualquier caso, conecta la placa a un puerto USB de tu orden

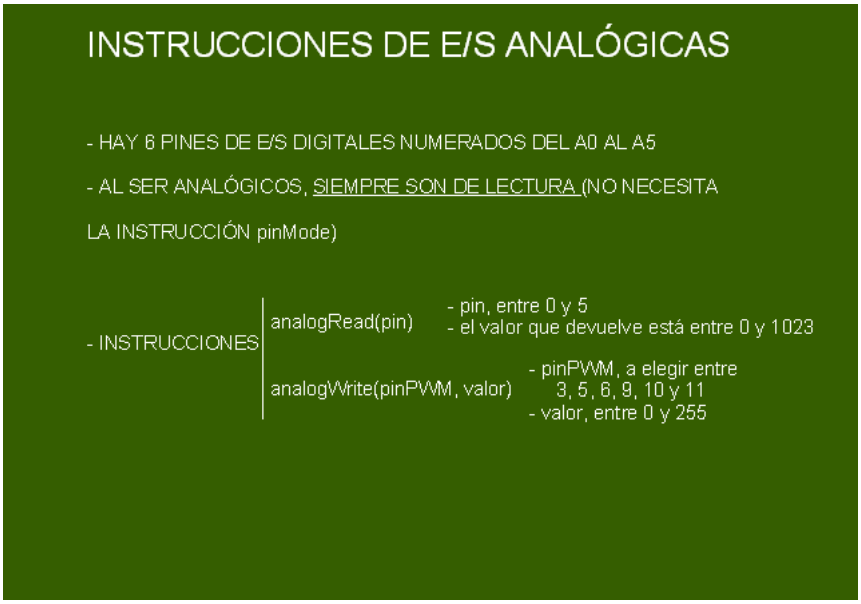


ESTRUCTURA DEL LENGUAJE DE PROGRAMACIÓN ARDUINO 1.- Organizado en al menos, dos funciones void setup() { Declaraciones; } voic loop() { Declaraciones; } 2.- Ambas son necesarias para que el programa funcione 3.-Se puede incluir una sección de declaración de funciones



EN UN PROGRAMA PUEDE HABER: - INSTRUCCIONES - FUNCIONES - DEFINICIÓN DE VARIABLES

INSTRUCCIONES DE E/S DIGITALES - HAY 14 PINES DE E/S DIGITALES NUMERADOS DEL 0 AL 13 - AL SER DIGITALES, LOS VALORES SE LEEN Y/O ESCRIBEN SON 0 Y 1 - INSTRUCCIONES - POSIBLES VALORES pinMode(pin,modo) digitalRead(pin) digitalWrite(pin, valor) - pin, entre 0 y 13 - modo, entre INPUT y OUTPUT - valor, entre HIGH y LOW



INSTRUCCIONES DE E/S ANALÓGICAS - HAY 6 PINES DE E/S DIGITALES NUMERADOS DEL A0 AL A5 - AL SER ANALÓGICOS, SIEMPRE DE LECTURA (NO NECESITA LA INSTRUCCIÓN pinMode) - INSTRUCCIONES analogRead(pin) analogWrite(pinPWM, valor) - pinPWM, a ele entre 3, 5, 6, 9, 10 y 11 - valor, entre 0 y 255 - pin, entre 0 y 5 - el valor que devuelve está entre 0 y 1023

PIN PWM (MODULACIÓN POR ANCHO DE PULSO) - HAY SEIS PINS PWM, 3, 5, 6, 9, 10 Y 11 - POSIBLES VALORES PARA LA INSTRUCCIÓN analogWrite(pinPWM, valor) Valor 0 -> salida de 0 voltios en el pin especificado; Valor de 255 -> salida de 5 voltios de salida en el pin especificad Valores entre 0 y 255, el pin saca tensiones entre 0 y 5 voltios - valor de 64 mantiene 0 voltios ¾ partes del tiempo y 5 voltios ¼ del tiempo; - valo 128 mantiene 0 voltios la mitad del tiempo y 5 voltios la otra mitad, - valor de 192 mantiene 0 voltios ¼ del tiempo y 5 voltios ¾ partes del tiempo

funciones

Una función es un bloque de código que tiene un nombre y un grupo de declaraciones que se ejecutan cuando se llama a la función. Podemos hacer uso de funciones integradas como *void setup()* y *void loop()* o escribir nuevas.

Las funciones se escriben para ejecutar tareas repetitivas y reducir el desorden en un programa. En primer lugar se declara el tipo de la función, que será el valor retornado por la función (*int, void...*). A continuación del tipo, se declara el nombre de la función y, entre paréntesis, los parámetros que se pasan a la función.

```
type functionName(parameters)
{
  statements;
}
```

La siguiente función *int delayVal()*, asigna un valor de retardo en un programa por lectura del valor de un potenciómetro.

```
int delayVal()
{
  int v;              //crea una variable temporal 'v'
  v = analogRead(pot); //lee el valor del potenciómetro
  v /= 4;             //convierte 0-1023 a 0-255
  return v;           //devuelve el valor final de v
}
```

EJEMPLO 1 DE PROGRAMA

```
int led = 10; //define el pin 10 como 'led'
int analog = 0;  // define el pin 0 como 'analog'
int valor;       // define la variable 'valor'

void setup() { //no es necesario configurar entradas y salidas

void loop()
{
valor =analogRead(analog);  //lee el pin 0 y lo asocia a
                            //la variable valor
valor /= 4; //divide valor entre 4 y lo reasigna a valor

analogWrite(led, value); // escribe en el pin10 valor
}
```

int led = 10; //define el pin 10 como 'led' int analog = 0; // define el pin 0 como 'analog' int valor; // define la variable 'valor' void setup() { //no e necesario configurar entradas y salidas void loop() { valor =analogRead(analog); //lee el pin 0 y lo asocia a //la variable valor valor /= 4; //divide entre 4 y lo reasigna a valor analogWrite(led, value); // escribe en el pin10 valor } EJEMPLO 1 DE PROGRAMA

ELEMENTOS DEL LENGUAJE

- llaves { }

```
void loop ()
  {
     Bloque de sentencias;
  }
```

- punto y coma ;
- bloques de comentarios /* */

```
/* este sería un ejemplo de
   comentario que continúa en
   otras líneas*/
```

- comentarios de una sola línea //

```
int contador = 0; //declaro la var contador y la inicializo a cero
```

EJEMPLO 2 DE PROGRAMA

```
int tiempo=200; int n; // Declaración inicial

void setup()
{ //comienza la configuracion
    for (n=5;n<9;n++)
    {
        pinMode (n, OUTPUT);
    }
}
/*A continuación defino una función llamada secuencia
Que será invocada por loop de manera continua */
void secuencia()
{
    for (n=5;n<9;n++)
    {
        digitalWrite (n, HIGH);     //activa el pin n
        delay (tiempo);             //cuenta 200 milisegundos
        digitalWrite (n, LOW);          //desactiva el pin n
        delay (tiempo);                 //cuenta 200 milisegundos
    }
}
void loop()
{
    secuencia();
}
```

int tiempo=200; int n; // Declaración inicial void setup() { //comienza la configuracion for (n=5;n< 9;n++) { pinMode (n, OUTPUT); } } /*A continuación defino una función llamada secuencia Que será invocada por loop de manera continua */ void secuencia() { for (n=5;n< 9;n++) { digitalWrite (n, HIGH); //activa el pin n delay (tiempo); //cuenta 200 milisegundos digitalWrite (n, LOW); //desactiva el pin n delay (tiempo); //cuenta 200 milisegundos } } void loop() { secuencia(); } EJEMPLO 2 DE PROGRAMA

- variables y declaración de variables
  int contador = 0;

- Tipos de variables:

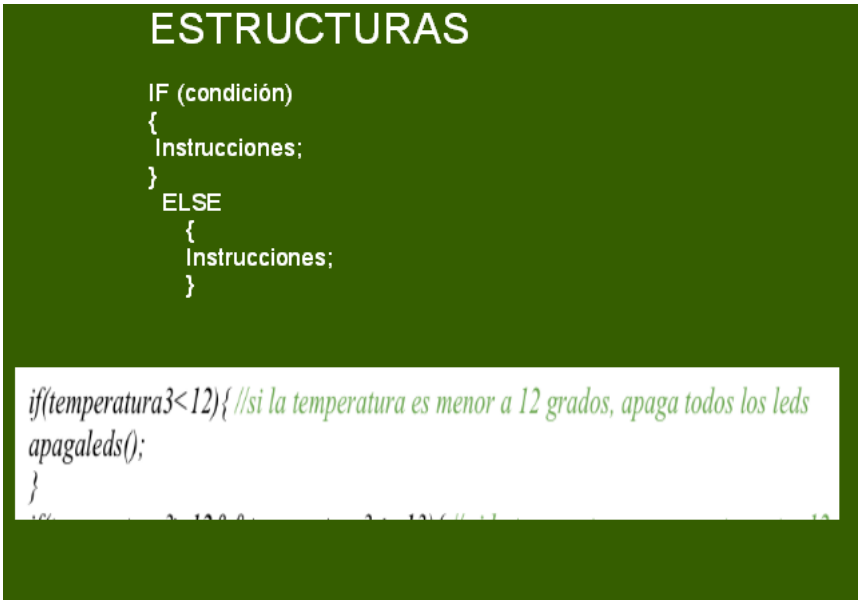
byte (entre 0 y 255 sin signo y sin coma decimal)
byte unaVariable=0;

int (entre –32767 y 32767, sin coma decimal, con signo.)
int saldoCuenta=0;

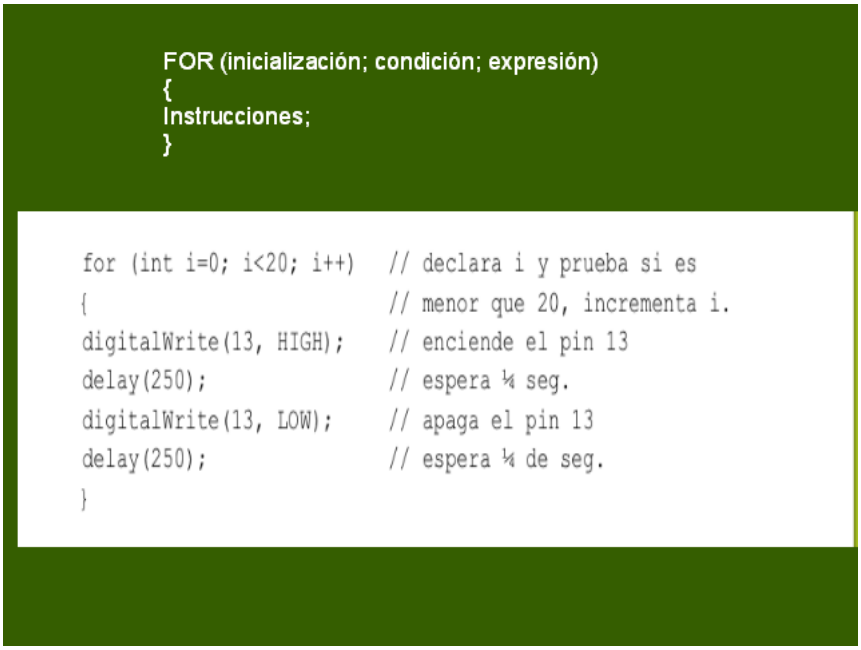
long (entre – 2147483647 y 2147483647 sin coma decimal)
long gastosGasolina= - 3541668;

float (entre – $3.4028235 \cdot 10^{38}$ y $3.4028235 \cdot 10^{38}$ con coma decimal)
float costeReparación= 354.16;

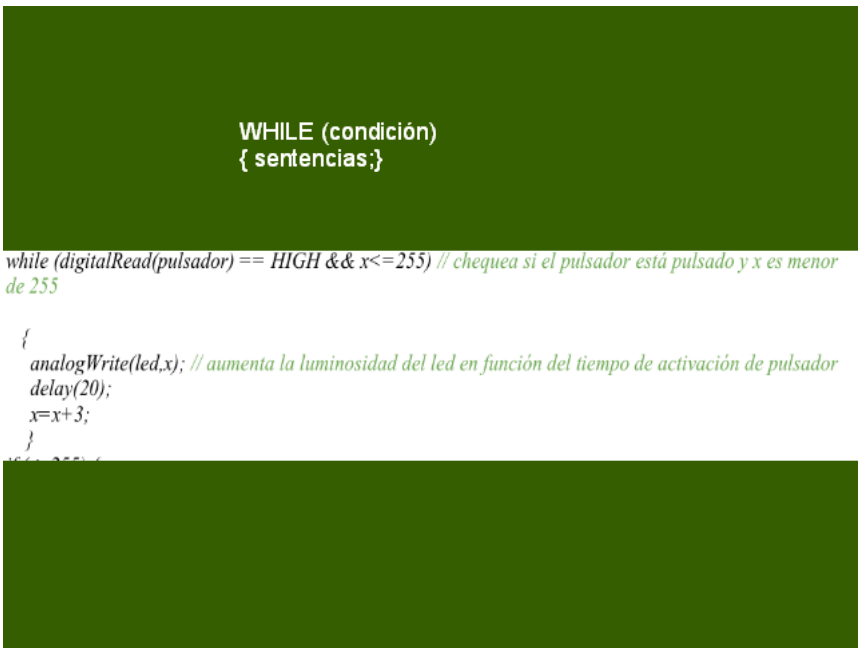
- variables y declaración de variables int contador = 0; - Tipos de variables: byte (entre 0 y 255 sin signo y sin coma decimal) byte unaVariable=0; (entre - 32767 y 32767, sin coma decimal, con signo.) int saldoCuenta=0; long (entre - 2147483647 y 2147483647 sin coma decimal) long gastosGasolina= - 3541668; float (entre - 3.4028235 · 1038 y 3.4028235 · 1038 con coma decimal) float costeReparación= 354.16;



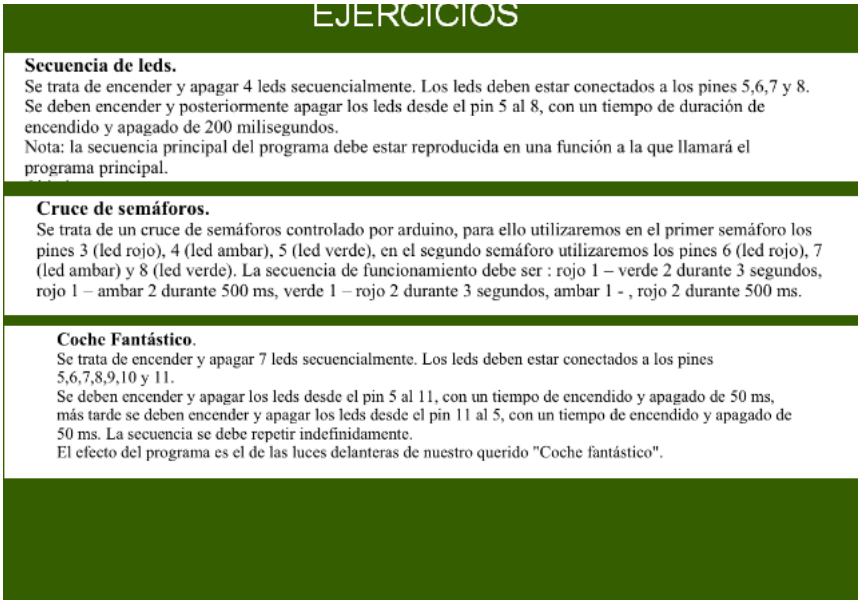
ESTRUCTURAS IF (condición) { Instrucciones; } ELSE { Instrucciones; }



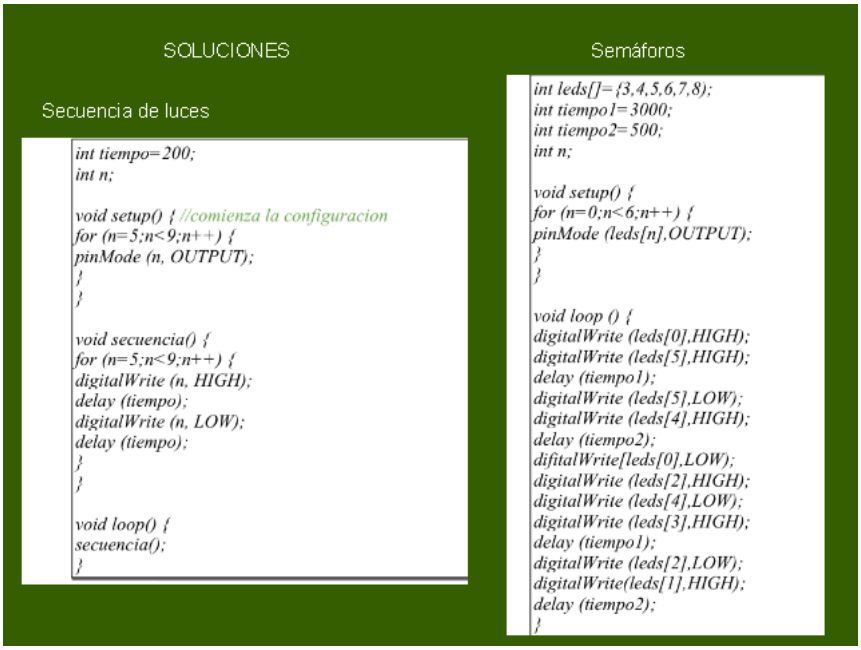
FOR (inicialización; condición; expresión) { Instrucciones; }



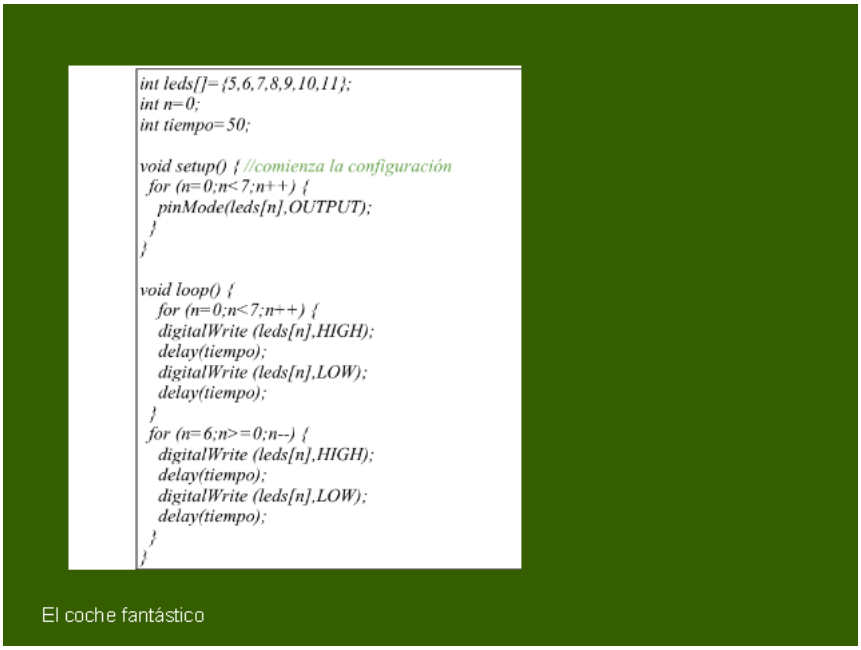
WHILE (condición) { sentencias;}



EJERCICIOS



SOLUCIONES Secuencia de luces Semáforos



El coche fantástico

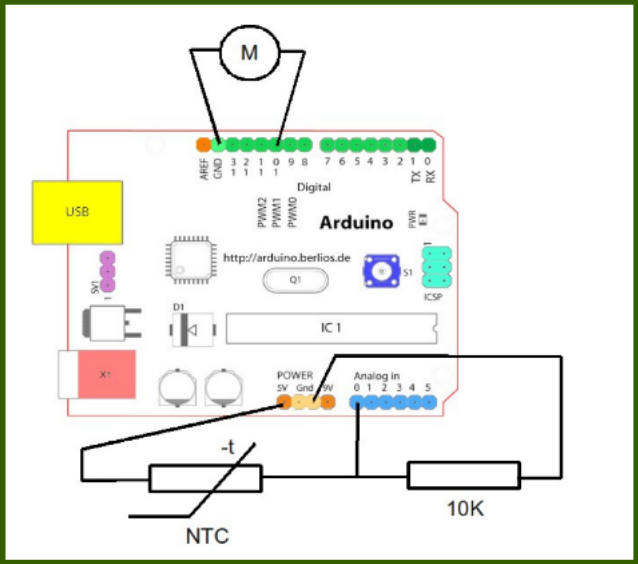
Aumentar y disminuir intensidad luminosa de led (fading).
Se trata aumentar y disminuir la luminosidad de un led usando la capacidad de ofrecer una tensión variable que da una salida analógica. Para ello se conecta un led al pin 11 y se provoca que su luminosidad pase de mínima a máxima, para luego ir de máxima a mínima. Los valores de salidas analógicas van del mínimo 0 al máximo 255.

```
int luminosidad = 0;  // variable para asignar la luminosidad al led
int led = 11; // pin del led

void setup()
{
  // en el setup no hay que configurar nada
}

void loop()
{
 for (luminosidad = 0 ; luminosidad <= 255; luminosidad=luminosidad+3) // fade in (from min to max)
  {
   analogWrite(led, luminosidad);  // ilumina el led con el valor asignado a luminosidad (entre 0 y 255)
   delay(30); // espera 30 ms para que se vea el efecto
  }
 for (luminosidad = 255; luminosidad >=0; luminosidad=luminosidad-3)   // fade out (from max to min)
  {
   analogWrite(led, luminosidad);
   delay(30);
  }
}
```

Ejemplo de divisor de tensión aplicado a un motor eléctrico

Termostato con velocidad de motor variable (Versión 2).
Se trata de un dispositivo que haga girar un motor más o menos rápido en función de la temperatura. Para ello conectaremos una ntc a la entrada analógica 0 y un led al pin 9 y el motor al pin 10. El valor de la entrada analógica 0 está comprendido entre 0 y 1024, y el valor del la tensión del pin 10 entre 0 y 5 voltios (entre 0 y 255). El motor debe girar a una velocidad entre 0 y 255 en función del valor de la entrada analógica 0, siendo su valor directamente proporcional al valor de la entrada analógica 0 (de 0 a 1024), o sea a más temperatura más velocidad del motor. Además el led del pin 9 debe encenderse.

Ejemplo de divisor de tensión aplicado a un motor eléctrico



(Esquema eléctrico)

(Esquema eléctrico)

```
int ntc=0;
int led=13;
int motor=9;
int n=0;
int temperatura=0;
int velocidadmotor=0;

void setup(){
 pinMode(led,OUTPUT);
 pinMode(motor,OUTPUT);
 Serial.begin(9600);
}

void monitoriza(){
 Serial.print("El valor de la temperatura es ...");
 Serial.println(temperatura);
 delay(1000);
}

void loop(){
 temperatura=analogRead(ntc);
 monitoriza();
 velocidadmotor=temperatura/4;
 digitalWrite(led,HIGH);
 analogWrite(motor,velocidadmotor);
}
```

Programa

Programa

PROGRAMACIÓN DE UN SERVO MOTOR

```
int servoPin = 2;   // servo conectado al pin digital 2
int myAngle;        // ángulo del servo de 0-180
int pulseWidth;     // anchura del pulso para la función
                    // servoPulse
void setup()
{
  pinMode(servoPin, OUTPUT);      // configura pin 2 como
}                                 // salida

void servoPulse(int servoPin, int myAngle)
{
  pulseWidth = (myAngle * 10) + 600;    // determina
                                        // retardo
  digitalWrite(servoPin, HIGH);   // activa el servo
  delayMicroseconds(pulseWidth); // pausa
  digitalWrite(servoPin, LOW);    // desactiva el servo
  delay(20);                      // retardo de refresco
}
void loop()   // el servo inicia su recorrido en 10° y
{             // gira hasta 170°
  for (myAngle=10; myAngle<=170; myAngle++)
  {
    servoPulse(servoPin, myAngle);
  }
              // el servo vuelve desde 170° hasta 10°
  for (myAngle=170; myAngle>=10; myAngle--)
  {
    servoPulse(servoPin, myAngle);
  }
}
```

PROGRAMACIÓN DE UN SERVO MOTOR

Comentarios

Para dejar un comentario, regístrese gratis o si ya está registrado, inicie sesión.

Trabajos relacionados

Estudio sobre los lenguajes de programación para la robótica
Origen de la palabra robot y su significado. Propiedades características de los robots. El robot y su funcionamiento. Cl...

Estructura de un objeto. Encapsulamiento y ocultación. Organización de los objetos. Actualmente una de las áreas más ca...

Sistemas de Procesamiento de Datos Programación Orientada a Objetos
Rupturas de Informe Definición de una Ruptura de Informe. Especificación de Opciones de Proceso. Una Ruptura de Informe se usa para dividir...

Ver mas trabajos de Programacion