



Visual programming environment for Arduino



PRÁCTICAS DE ARDUINO / VISUALINO

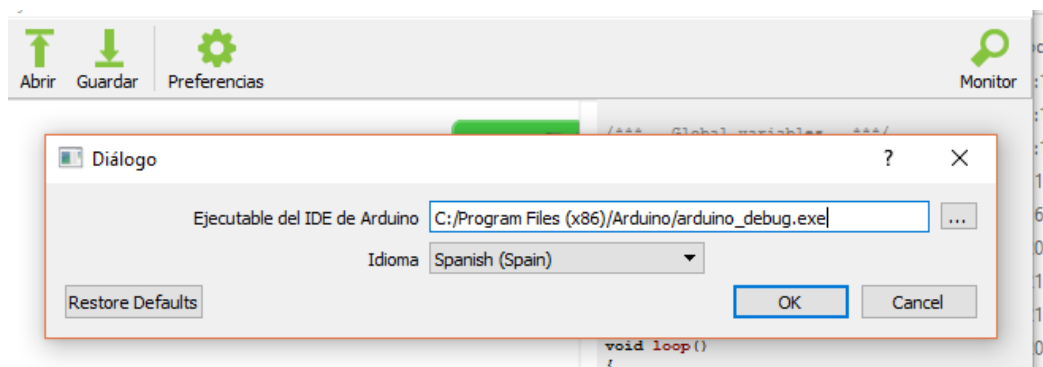
Prof. Vicente Marqués García DTO. TECNOLOGÍA IES COLONIAL

ÍNDICE

1. INTRODUCCIÓN, INSTALACIÓN Y CONFIGURACIÓN	2
2. PRÁCTICAS	2
2.1. Intermitente	2
2.2. Alarma-1	5
2.3. Secuencia Básica de 3 LEDs	7
2.4. Alarma-2. Condición con variable.....	9
2.5. Contador.	11
2.6. Entrada Analógica.....	13
2.7. Sensor de Luz o LDR (Light Dependent Resistor)	15
2.8. Sensor de temperatura o NTC	17
2.9. Control de un motor de cc con un transistor.....	19
2.10. Control del sentido de giro de motor de CC.....	22
2.11. Control de un dispositivo de C.A. con un relé.....	24
2.12. Medir distancias con el sensor de ultrasonidos.....	26
2.13. Contador de tiempo y visualización en LCD 16x2	29
3. ESTACIÓN METEOROLÓGICA	31
3.1. Programación la Estación Meteorológica en Visualino/IDE de Arduino	32

1. INTRODUCCIÓN, INSTALACIÓN Y CONFIGURACIÓN

1. Instalar Software de Arduino.
2. Instalar Software de Visualino. Requiere que esté instalada la última versión del IDE de Arduino.
3. Configurar las Preferencias de Visualino para que esté localizado.

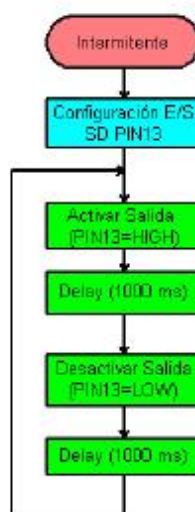


2. PRÁCTICAS

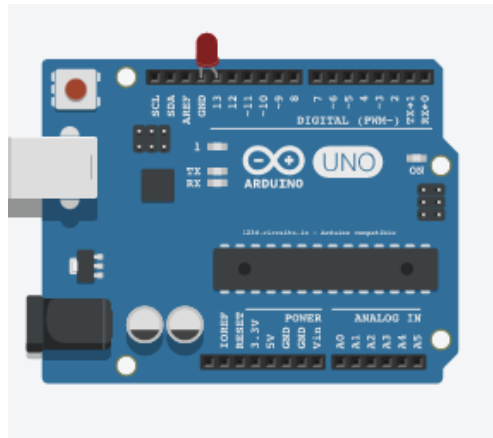
2.1. Intermitente

Se trata de realizar un ejercicio básico que consiste en encender y a pagar un led que conectamos en el PIN 13 de Arduino que lo configuramos como salida. El tiempo de encendido y apagado es de 1 segundo.

Organigrama y Esquema

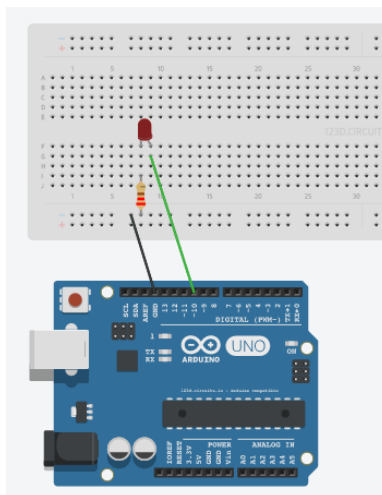


Organigrama y esquema de conexionado con la tarjeta Arduino.



Obsérvese que se ha colocado el diodo led sin resistencia en serie dado que el PIN13 de Arduino ya lleva incorporada una resistencia interior. En el caso de colocar el diodo LED en otra salida deberíamos colocar una resistencia de al entre 220 y 500 ohmios dependiendo del consumo de corriente del diodo.

Conexionado a realizar en el caso de realizar la salida por el PIN 10



Listado de componentes:

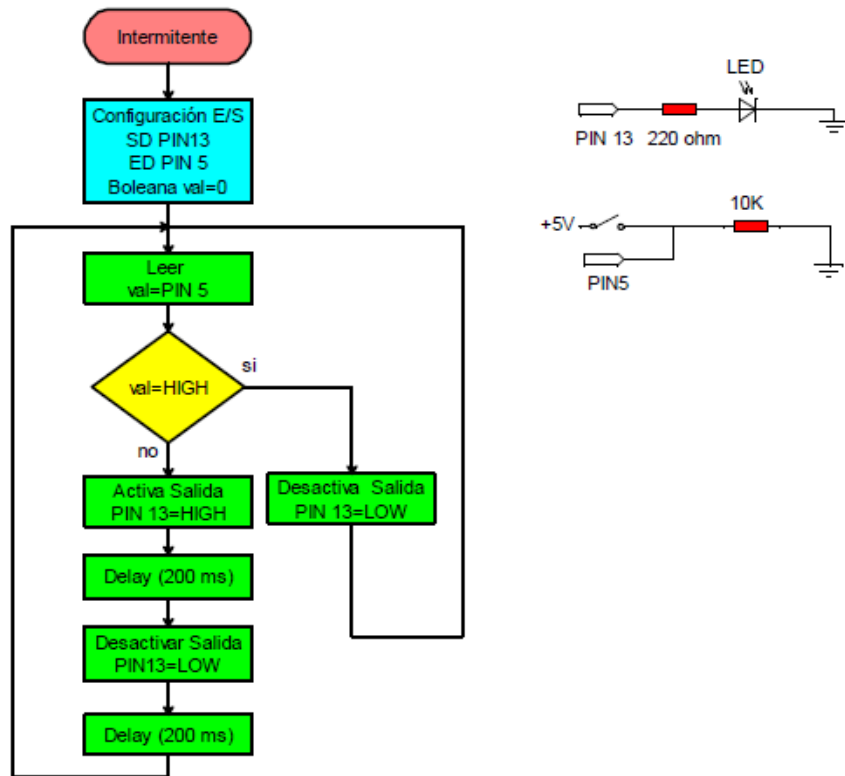
- 1 Resistencia 220 Ω
- 1 Diodo LED

PROGRAMA VISUALINO**PROGRAMA**

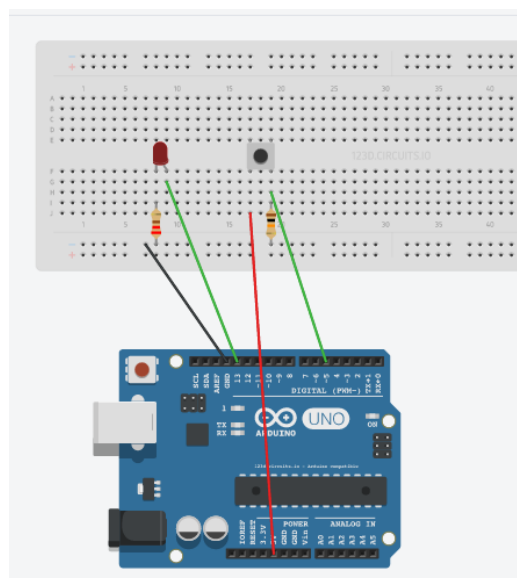
```
void setup()
{
  pinMode(13,OUTPUT);
}
void loop()
{
  digitalWrite(13,HIGH);
  delay(1000);
  digitalWrite(13,LOW);
  delay(1000);
}
```

2.2. Alarma-1

Cuando se pulsa el pulsador (Pin5 a "0") se enciende y se apaga de forma intermitente el pin 13.



MONTAJE



Listado de componentes:

- 1 pulsador
- 1 Resistencia 10 k Ω
- 1 Resistencia 220 Ω
- 1 Diodo LED

PROGRAMA VISUALINO



PROGRAMA

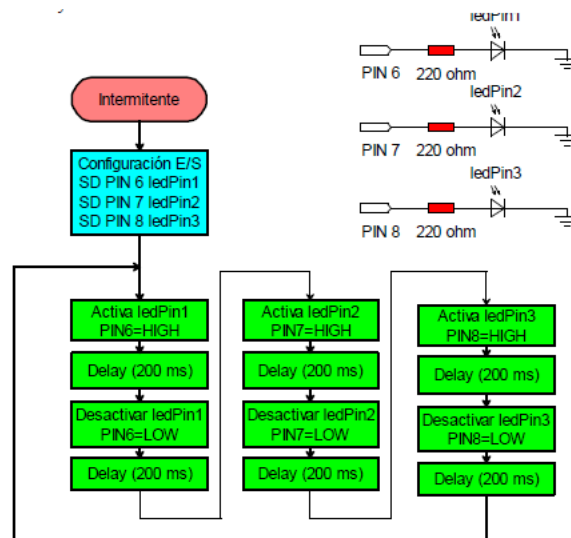
```

void setup()
{
  pinMode(5,INPUT);
  pinMode(13,OUTPUT);
}
void loop()
{
  if (digitalRead(5) == HIGH) {
    digitalWrite(13,HIGH);
    delay(200);
    digitalWrite(13,LOW);
    delay(200);
  }
}

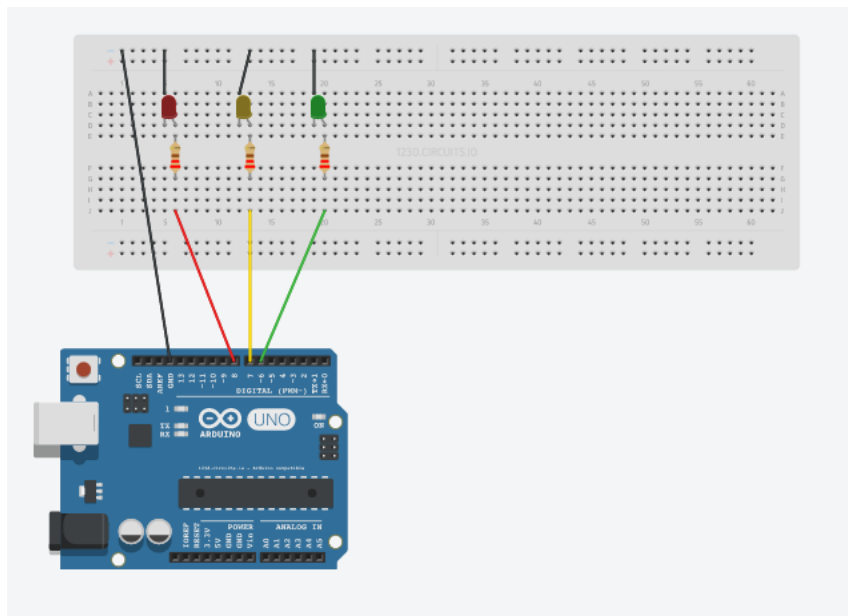
```

2.3. Secuencia Básica de 3 LEDs

Se trata de encender y apagar 3 LEDs colocados en las salidas 6, 7 y 8 (PIN6, PIN7 y PIN8) con una cadencia de 200 ms. Las variables asignadas a cada led son ledPin1, ledPin2 y ledPin3.



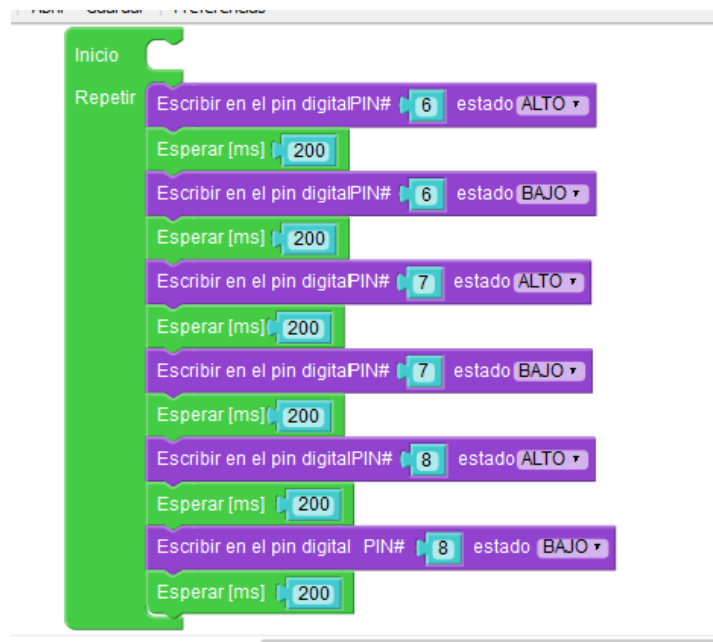
MONTAJE



Listado de componentes:

- 3 Resistencia 220 Ω
- 3 Diodo LED

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```

void setup()
{
  pinMode(6,OUTPUT);
  pinMode(7,OUTPUT);
  pinMode(8,OUTPUT);
}
void loop()
{
  digitalWrite(6,HIGH);
  delay(200);
  digitalWrite(6,LOW);
  delay(200);
  digitalWrite(7,HIGH);
  delay(200);
  digitalWrite(7,LOW);
  delay(200);
  digitalWrite(8,HIGH);

```

```

delay(200);
digitalWrite(8,LOW);
delay(200);
}

```

RETO FINAL

Modifica el programa para que el led rojo parpadee 2 segundos, el verde 3 segundos y el verde 4.

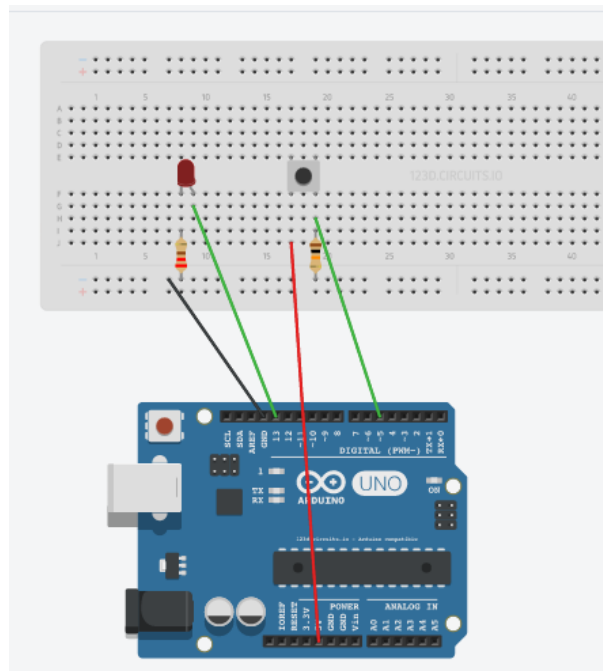
2.4. Alarma-2. Condición con variable

En esta práctica vamos a leer el valor de una entrada y lo vamos a guardar en una variable global llamada "Valor_Pulsador".

Para ello primero tenemos que declararla. Posteriormente la alarma se activará o no en función del valor que tenga "Valor-Pulsador".

El montaje es idéntico al de la alarma 1.

Aclaraciones: Para que aparezca "en cambio si" en el bloque si, es necesario picar sobre * de color azul y arrastrarlo.



Listado de componentes:

- 1 pulsador
- 1 Resistencia 10 k Ω
- 1 Resistencia 220 Ω
- 1 Diodo LED

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```

/**/ Global variables */
int Valor_Pulsador=0;
/**/ Function declaration */
void setup()
{
  pinMode(5,INPUT);
  pinMode(13,OUTPUT);
}
void loop()
{
  Valor_Pulsador=digitalRead(5);
  if (Valor_Pulsador == 1) {
    digitalWrite(13,HIGH);
  }
}

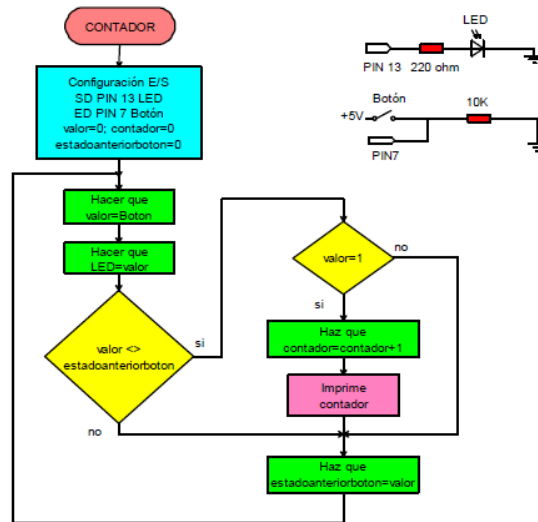
```

```

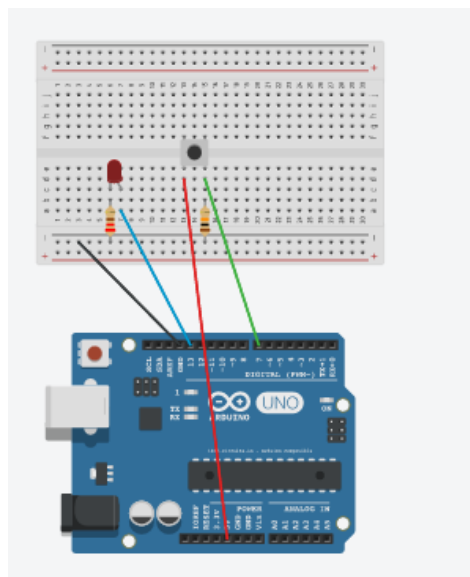
}else if (Valor_Pulsador == 0) {
    digitalWrite(13,LOW);
}
}
}
    
```

2.5. Contador.

Se trata de contar las veces que se pulsa un botón conectado en la entrada 7 de Arduino a la vez que cada vez que contamos encendemos el led conectado en la salida 13. El valor de la variable que almacena el número de impulsos generados se envía al PC para que se visualice en la pantalla.



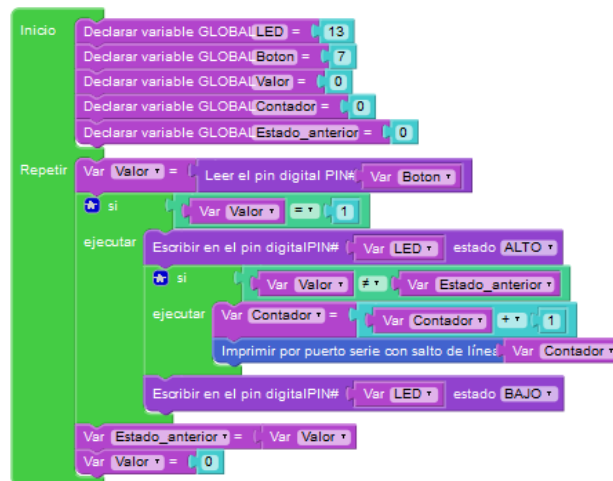
MONTAJE



Listado de componentes:

- 1 pulsador
- 1 Resistencia 10 k Ω
- 1 Resistencia 220 Ω
- 1 Diodo LED

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```

/** Global variables */
int LED=13;
int Boton=7;
int Valor=0;
int Contador=0;
int Estado_anterior=0;
/** Function declaration */
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  pinMode(Boton,INPUT);
  Valor=digitalRead(Boton);

```

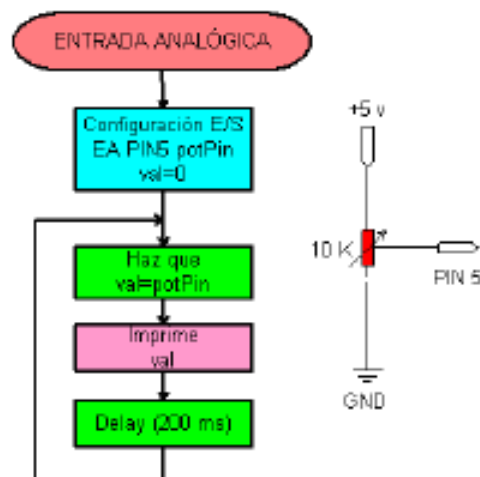
```

if (Valor == 1) {
  pinMode(LED,OUTPUT);
  digitalWrite(LED,HIGH);
  if (Valor != Estado_anterior) {
    Contador=Contador + 1;
    Serial.println(Contador);
  }
  pinMode(LED,OUTPUT);
  digitalWrite(LED,LOW);
}
Estado_anterior=Valor;
Valor=0;
}

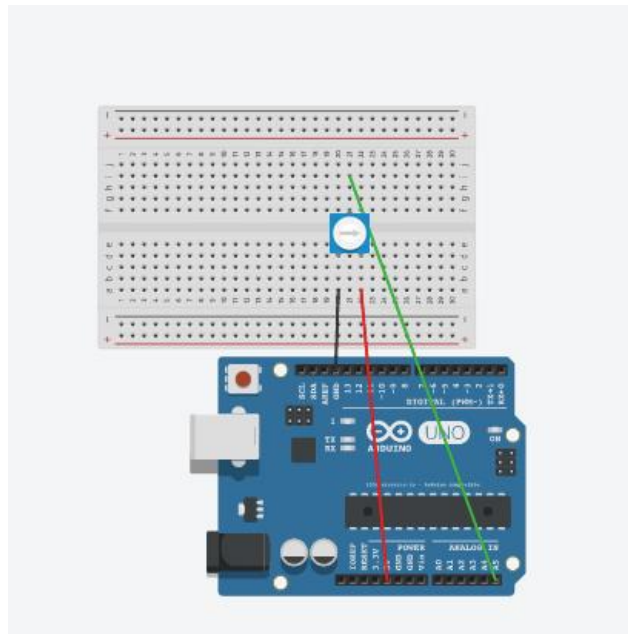
```

2.6. Entrada Analógica

Se trata de configurar un canal de entrada analógico (pin A5) y enviar el valor leído al PC para visualizarlo.



MONTAJE



Listado de componentes:

- 1 potenciómetro

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```

/** Global variables */
int Pin_Potenciometro=5;
int Valor=0;
    
```

```

/**** Function declaration ****/
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  pinMode(Pin_Potenciometro,INPUT);
  Valor=analogRead(Pin_Potenciometro);
  Serial.println(Valor);
  delay(200);
}

```

2.7. Sensor de Luz o LDR (Light Dependent Resistor)

Un LDR es una resistencia variable, que varía su valor dependiendo de la cantidad de luz que incide sobre su superficie. Cuanta más intensidad de luz incide en la superficie de la LDR, menor será su resistencia, y cuanto menos luz mayor será la resistencia. Suelen ser utilizados como sensores de luz ambiental o como una fotocélula que activa un determinado proceso en ausencia o presencia de luz.

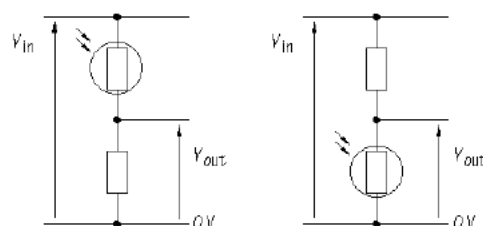
Un sensor de luz se compone de una LDR como parte de un divisor de tensión resistivo.

Ejemplo:

$$V_{out} = \left(\frac{R_{bottom}}{R_{bottom} + R_{top}} \right) * V_{in}$$

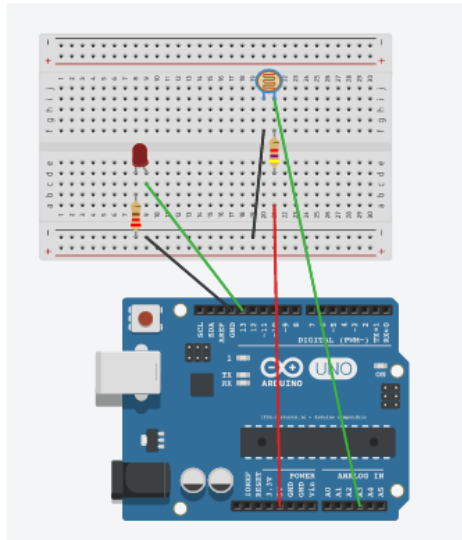
Si la LDR es usada como R_{top} , como en el primer circuito, da tensión alta (HIGH) en la salida cuando la LDR está en la luz, y una tensión baja (LOW) en la salida cuando la LDR está en la sombra.

La acción del divisor de tensión es inversa cuando la LDR es usada como R_{bottom} en lugar de R_{top} , como en el segundo circuito. El circuito da tensión Baja (LOW) en la salida cuando la LDR está en la luz, y una tensión alta (HIGH) en la salida cuando la LDR está en la sombra. El circuito divisor de tensión dará una tensión de la salida que cambia con la iluminación, de forma inversamente proporcional a la cantidad de luz que reciba (sensor de oscuridad).



En esta práctica haremos un programa que lea el valor de V_{out} por el pin analógico A3 y muestre dicha lectura por monitor serie.

MONTAJE



Listado de componentes:

- 1 LDR sensor de luz
- 1 Resistencia 5 k Ω
- 1 Resistencia 220 Ω
- 1 Diodo LED

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```
/** Global variables */
int Light_Pin=3;
int Led_Pin=13;
int Valor=0;
/** Function declaration */
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  pinMode(Light_Pin,INPUT);
  Valor=analogRead(Light_Pin);
  Serial.println(Valor);
  delay(200);
}
```

RETO FINAL

Modificar el montaje y el programa para que se encienda un led conectado al pin 13 en caso de que la lectura obtenida de la LDR sea inferior a 500.

(Si no lo consigues pasa a la práctica siguiente)

2.8. Sensor de temperatura o NTC

En este ejemplo se trata de medir la temperatura desde el A3 de entrada analógica y ver si este valor supera un valor dado de 500 (medida absoluta) si supera este valor activará la salida digital PIN13 y si no la apagará. Además queremos que se muestre en el monitor de salida del IDE Arduino el valor leído. D sensor utilizaremos un sensor del tipo NTC.

Un NTC o termistor NTC es una resistencia variable, que varía su valor dependiendo de la temperatura ambiente. Cuanta mas temperatura menor será su resistencia y cuanto menos temperatura, mayor será la resistencia. Suelen ser utilizados en alarmas.

Un sensor de temperatura se compone de un NTC como parte de un divisor de tensión resistivo.

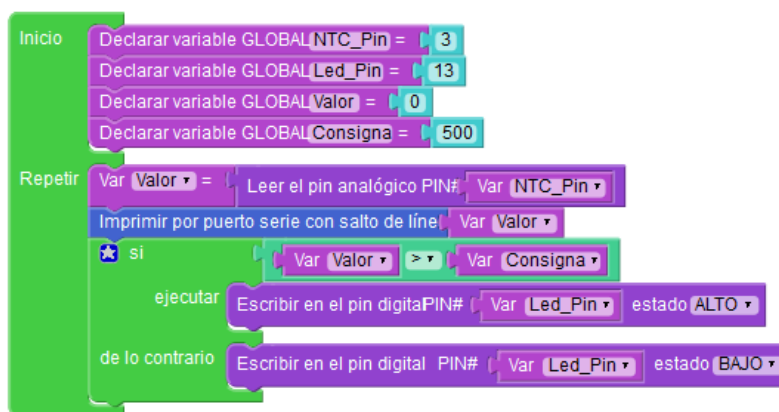
MONTAJE

El montaje es idéntico al de la práctica anterior cambiando la LDR por una NTC.

Listado de componentes:

- 1 NTC sensor de temperatura
- 1 Resistencia 5 k Ω
- 1 Resistencia 220 Ω
- 1 Diodo LED

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```

/** Global variables */
int NTC_Pin=3;
int Led_Pin=13;
int Valor=0;
int Consigna=500;
/** Function declaration */
void setup()
{
  Serial.begin(9600);

```

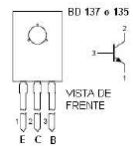
```

}
void loop()
{
  pinMode(NTC_Pin,INPUT);
  Valor=analogRead(NTC_Pin);
  Serial.println(Valor);
  if (Valor > Consigna) {
    pinMode(Led_Pin,OUTPUT);
    digitalWrite(Led_Pin,HIGH);
  }else {
    pinMode(Led_Pin,OUTPUT);
    digitalWrite(Led_Pin,LOW);
  }
}
}

```

2.9. Control de un motor de cc con un transistor

Con este ejemplo vamos a controlar la velocidad de un motor de cc mediante la utilización de un transistor BD137. Se trata de utilizar la posibilidad de enviar una señal de PWM a una de las salidas configurables como salidas analógicas.



Téngase en cuenta que el motor debe ser de bajo consumo por dos motivos: primero porque si alimentamos en las pruebas desde el conector USB no debemos sacar demasiada corriente del ordenador y segundo porque el transistor es de una corriente limitada.

El diodo 1N4001 se coloca como protección para evitar que las corrientes inversas creadas en el bobinado del motor puedan dañar el transistor.

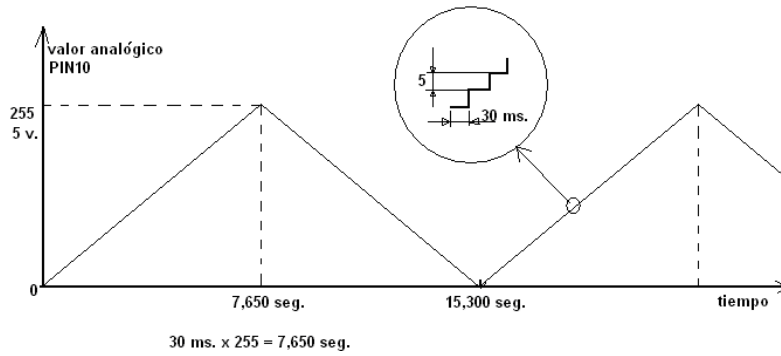
Prácticas con Arduino Nivel I

64

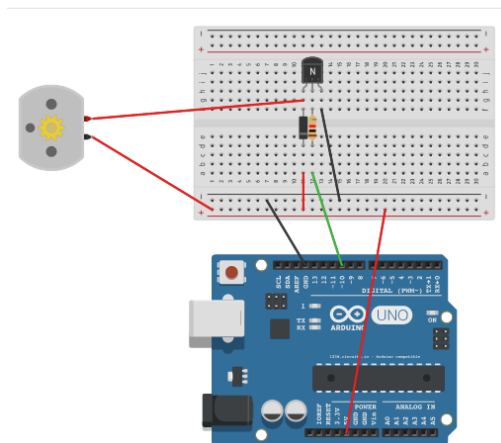
La tensión que sacaremos a la salida 10 (analógica tipo PWM) variara en forma de rampa ascendente y descendente de manera cíclica tal como vemos en la figura. Este efecto lo conseguimos con una estructura del tipo for:

- for(valor = 0 ; valor <= 255; valor +=1) (ascendente)
- for(valor = 255; valor >=0; valor -=1) (descendente)

Obsérvese que los incrementos del valor de la tensión van de 1 en 1 y tenemos que considerar que 0 V equivale a 0 y 5 V equivale a 255.



MONTAJE



NOTA: El transistor que aparece en el montaje tiene las patillas en el siguiente orden: C, B, E. Por tanto difiere del transistor BD 137 (E, C; B) que utilizaremos en el montaje y habrá que tener en cuenta dicha circunstancia para hacer el montaje correctamente.

Listado de componentes:

- 1 Transistor BD 137
- 1 Diodo 1N4001
- 1 Resistencia 1 K Ω
- 1 Potenciómetro (RETO FINAL)

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```

/** Global variables */
int Pin_Motor=10;
int Valor=0;
/** Function declaration */
void setup()
{
  Serial.begin(9600);
}
void loop()
{
  for (Valor = 0; Valor <= 255; Valor++) {
    pinMode(Pin_Motor,OUTPUT);
    analogWrite(Pin_Motor,Valor);
    Serial.println(Valor);
    delay(30);
  }
  for (Valor = 255; Valor >= 0; Valor--) {
    pinMode(Pin_Motor,OUTPUT);
    analogWrite(Pin_Motor,Valor);
    Serial.println(Valor);
  }
}

```

```

    delay(30);
  }
}

```

RETO FINAL

Conecta un potenciómetro por un pin analógico (práctica 6) para hacer que el motor varíe la velocidad según giremos el potenciómetro.

2.10. Control del sentido de giro de motor de CC

Para controlar el sentido de giro de un motor de corriente continua utilizaremos el módulo Driver dual para motores. [MÁS INFO](#)

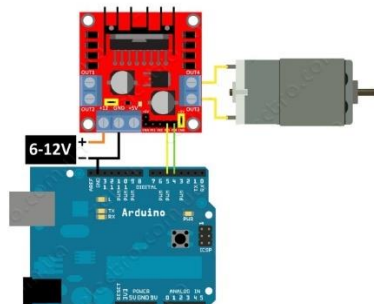
Conectaremos un pulsador con una resistencia de 10 K Ω para tomar valores 0 / 1 por el pin 2. En función de si está pulsado (1) o no (0) girará en un sentido u otro.

Para ello activaremos dos salidas digitales conectadas a los pines 4 y 5. La activación de cada sentido de giro se corresponde con la siguiente tabla:

PIN DIGITAL ARDUINO	PIN MÓDULO H	VALORES	MOTOR
4	IN 4	1	Gira en un sentido
5	IN 3	0	
4	IN 4	0	Motor parado
5	IN 3	0	
4	IN 4	1	Gira en sentido contrario
5	IN 3	0	

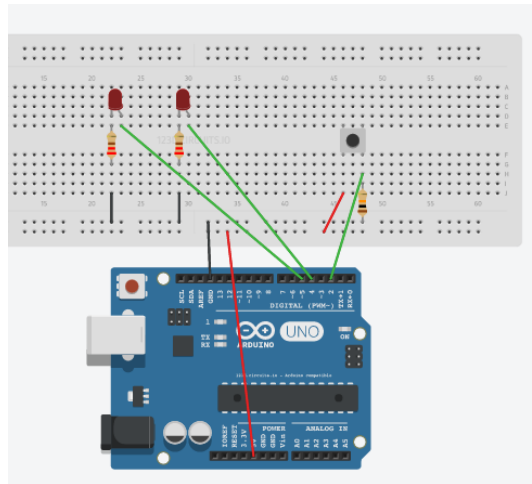
Debemos tener habilitada la entrada correspondiente según la información mostrada en el anterior link.

MONTAJE



Para poder hacer el programa con visualino, dado que no dispone del puente H, utilizaremos dos leds conectados a los pines 4 y 5. De esta forma podremos simular el resultado del programa.

En el montaje real el Pin 4 de arduino se conectará con la entrada de control IN4 del puente H y el PIN 5 con la entrada de control IN5.



PROGRAMA VISUALINO



PROGRAMA ARDUINO

```
/** Global variables */
```

```
int Pin_Pulsador=2;
```

```
int Valor_Pulsador=0;
```

```
int Pin_IN4=4;
```

```
int Pin_IN3=5;
```

```
/** Function declaration */
```



```

void setup()
{
}

void loop()
{
  pinMode(Pin_Pulsador,INPUT);
  Valor_Pulsador=digitalRead(Pin_Pulsador);
  if (Valor_Pulsador == true) {
    pinMode(Pin_IN4,OUTPUT);
    digitalWrite(Pin_IN4,HIGH);
    pinMode(Pin_IN3,OUTPUT);
    digitalWrite(Pin_IN3,LOW);
  }else {
    pinMode(Pin_IN4,OUTPUT);
    digitalWrite(Pin_IN4,LOW);
    pinMode(Pin_IN3,OUTPUT);
    digitalWrite(Pin_IN3,HIGH);
  }
}
}

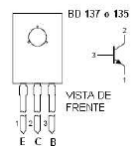
```

2.11. Control de un dispositivo de C.A. con un relé

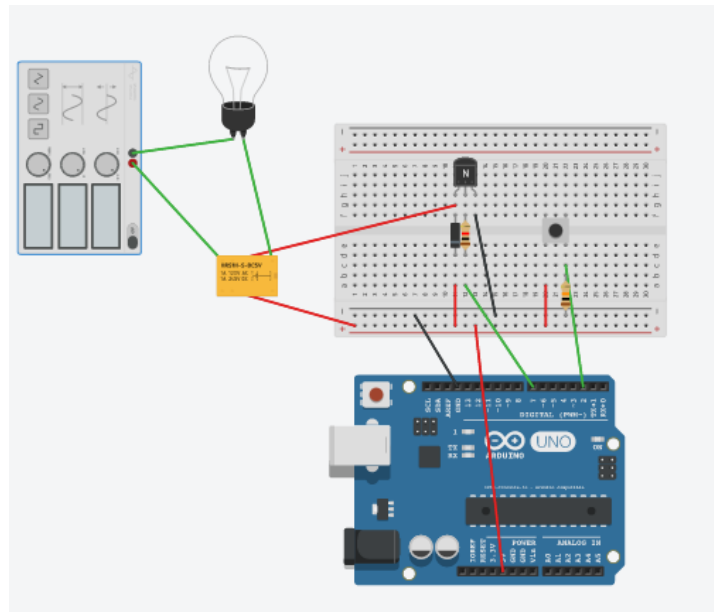
Como en la práctica 9, utilizaremos un transistor BD137, un diodo y un relé. En este caso lo controlaremos mediante una salida digital.

El relé deberá activarse con la salida digital de 5V de corriente continua y admitirá la potencia del dispositivo a conectar.

El circuito dispone de un pulsador que al pulsarlo activa el relé y conecta el equipo de 230 V. Al soltarlo se apagará dicho equipo.



MONTAJE



NOTA: El transistor que aparece en el montaje tiene las patillas en el siguiente orden: C, B, E. Por tanto difiere del transistor BD 137 (E, C; B) que utilizaremos en el montaje y habrá que tener en cuenta dicha circunstancia para hacer el montaje correctamente.

Listado de componentes:

- 1 pulsador
- 1 resistencia 10 K Ω
- 1 Transistor BD 137
- 1 Diodo 1N4001
- 1 Resistencia 1 K Ω
- 1 Relé para 5V y la potencia dependerá del elemento a controlar de corriente alterna.

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```
/** Global variables */
int Pin_Pulsador=2;
int Valor_Pulsador=0;
int Pin_Rele=7;

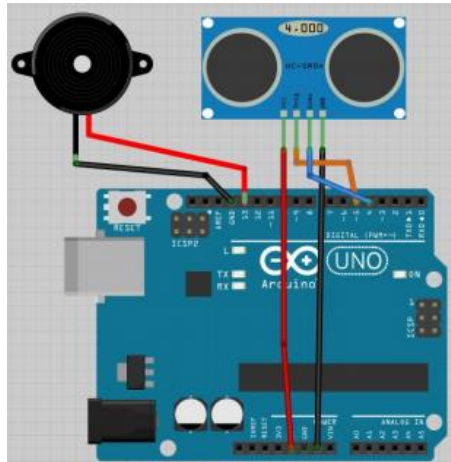
/** Function declaration */
void setup()
{
}
void loop()
{
  pinMode(Pin_Pulsador,INPUT);
  Valor_Pulsador=digitalRead(Pin_Pulsador);
  if (Valor_Pulsador == true) {
    pinMode(Pin_Rele,OUTPUT);
    digitalWrite(Pin_Rele,HIGH);
  }else {
    pinMode(Pin_Rele,OUTPUT);
    digitalWrite(Pin_Rele,LOW);
  }
}
```

2.12. Medir distancias con el sensor de ultrasonidos

El sensor de ultrasonidos tiene cuatro conexiones. Tensión de alimentación o Vcc que hay que conectar a 5V, tierra o GND, ECHO y TRIGGER. La combinación de los dos últimos nos da una distancia a un obstáculo que se encuentre enfrente de nosotros.

Hacer un programa que "pite" y se encienda un led conectado al pin 12 cuando se acerque un obstáculo a menos de 30cm de nosotros. [MÁS INFO](#)

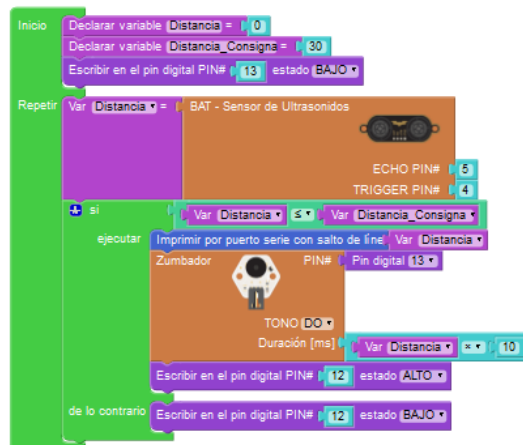
MONTAJE



Listado de componentes:

- 1 sensor de ultrasonidos SR-4
- 1 resistencia 220 Ω
- 1 LED
- 1 Zumbador

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```
/** Global variables */
```

```
/** Function declaration */
```

```
//bqBAT
long TP_init(int trigger_pin, int echo_pin);
long Distance(int trigger_pin, int echo_pin);

void setup()
{
  pinMode(13,OUTPUT);

  int Distancia=0;
  int Distancia_Consigna=30;
  digitalWrite(13,LOW);

  pinMode( 5 , INPUT );
  pinMode( 4 , OUTPUT );
  Serial.begin(9600);
  pinMode(12,OUTPUT);
}
void loop()
{
  Distancia=Distance(4,5);
  if (Distancia <= Distancia_Consigna) {
    Serial.println(Distancia);
    tone(13,261,Distancia * 10);
    delay(Distancia * 10);
    digitalWrite(12,HIGH);
  }else {
    digitalWrite(12,LOW);
  }
}
}
/** Function definition */
//bqBAT
long TP_init(int trigger_pin, int echo_pin)
{
  digitalWrite(trigger_pin, LOW);
  delayMicroseconds(2);
```

```

digitalWrite(trigger_pin, HIGH);
delayMicroseconds(10);
digitalWrite(trigger_pin, LOW);
long microseconds = pulseIn(echo_pin ,HIGH);
return microseconds;
}
long Distance(int trigger_pin, int echo_pin)
{
long microseconds = TP_init(trigger_pin, echo_pin);
long distance;
distance = microseconds/29/2;
if (distance == 0){
distance = 999;
}
return distance;

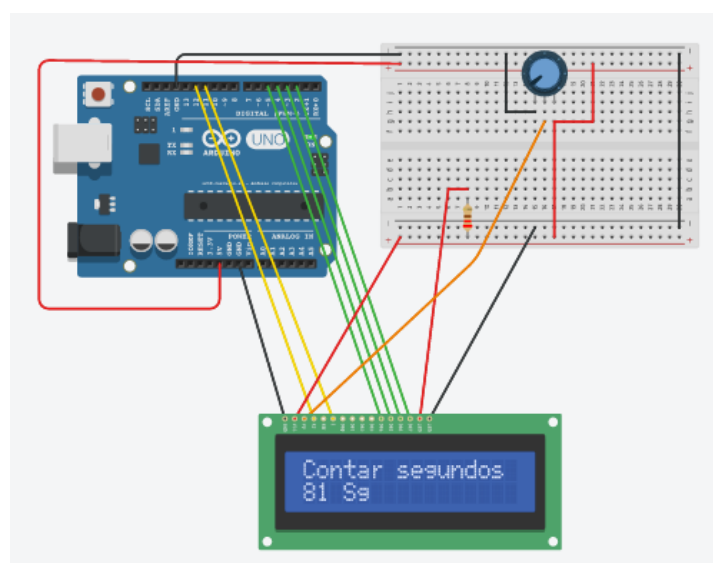
```

2.13. Contador de tiempo y visualización en LCD 16x2

En esta práctica contaremos segundos y mostraremos dicho contaje en un LCD 16x2.

El bloque AJUSTAR RETROILUMINACIÓN está obsoleto y No funcionará. Los desarrolladores de VISUALINO contemplan que este bloque tendrá en un futuro la posibilidad de cambiar los pines de conexionado. Por ahora, sólo acepta la combinación para la librería LiquidCrystal 12, 11, 5, 4, 3 y 2.

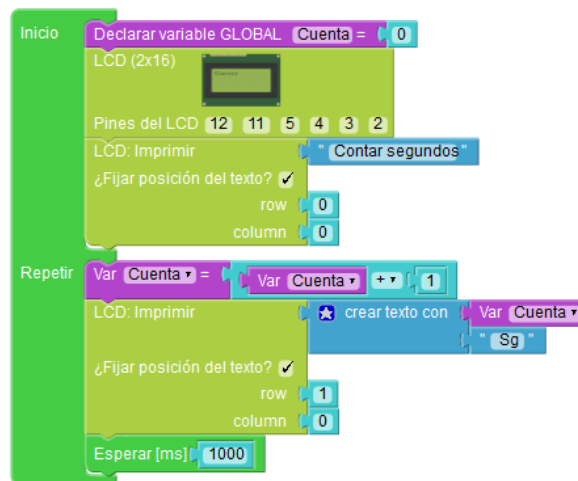
MONTAJE



Listado de componentes:

- 1 LCD 16x2
- 1 resistencia 220 Ω
- 1 potenciómetro 10 K Ω

PROGRAMA VISUALINO



PROGRAMA ARDUINO

```

#include <Wire.h>
#include <LiquidCrystal.h>

/**/ Global variables ***/
int Cuenta=0;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

/**/ Function declaration ***/

void setup()
{
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0,0);

```

```
lcd.print("Contar segundos");  
}  
void loop()  
{  
  Cuenta=Cuenta + 1;  
  lcd.setCursor(0,1);  
  lcd.print(String(Cuenta) + String(" Sg"));  
  delay(1000);  
}
```

RETO FINAL

Realiza un reloj que muestre en pantalla la hora según el siguiente formato: hh:mm:ss

3. ESTACIÓN METEOROLÓGICA

Ya puedes construir una estación meteorológica para medir temperatura, humedad y pluviometría.

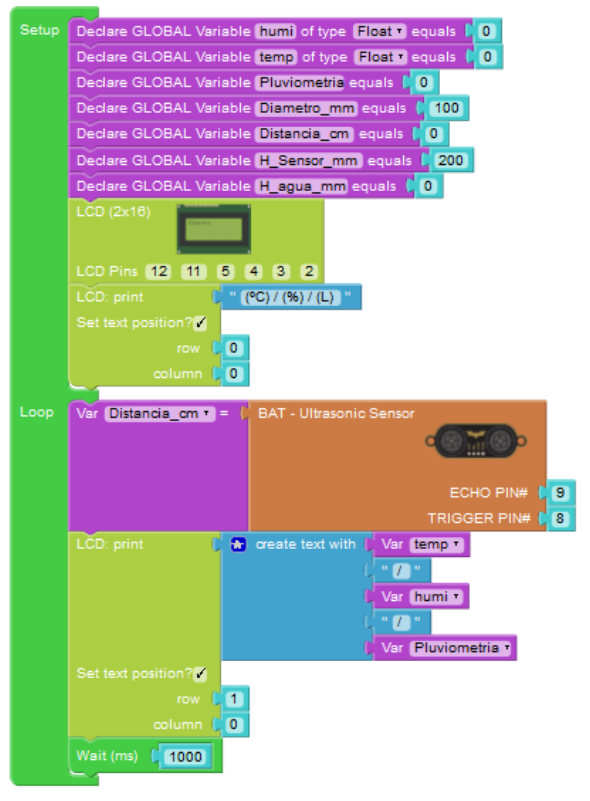
Para ello deberás utilizar un sensor DTH11, una pantalla LCD y un sensor de ultrasonidos.

En el siguiente enlace tienes más información para realizar dicho proyecto: [Aquí](#)

En el siguiente punto puedes ver como desarrollar el programa utilizando visualino y el IDE de Arduino.

3.1. Programación la Estación Meteorológica en Visualino/IDE de Arduino

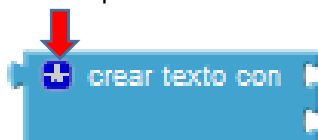
PROGRAMA CON VISUALINO (LCD, SENSOR ULTRASONIDOS)



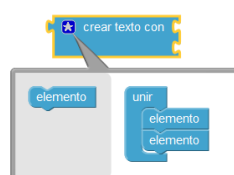
Observaciones:

1. Es muy **IMPORTANTE** que las variables para humedad y temperatura se llamen respectivamente **humi** y **temp**.
2. Cuidado que las variables humi y temp son de **tipo decimal**.
3. Para crear texto con más de 3 entradas:

En el bloque crear texto

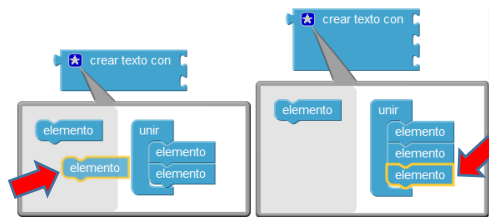


Hacer click sobre el icono que indica la flecha roja



Por defecto aparecen 2 elementos dentro del bloque unir.

Para obtener más entradas, hay que arrastrar nuevos elementos desde el lado de la izquierda:



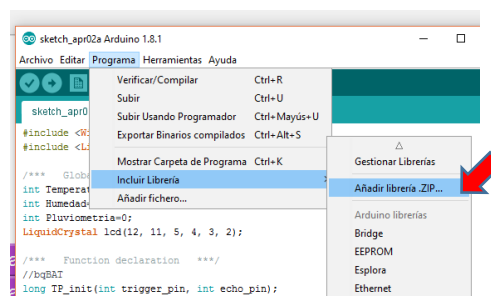
MODIFICACIÓN DEL PROGRAMA EN EL IDE DE ARDUINO (SENSOR DHT11)

A continuación, modificamos el programa con el IDE de arduino para incluir la medida de Humedad, Temperatura y calcular la pluviometría según la medida de distancia:

Descargamos la librería del sensor DHT-11: [Aquí](#)

<http://www.prometec.net/wp-content/uploads/2014/10/DHT11.zip>

Incluimos la librería en el IDE en Programa/Incluir Librería/Añadir librería.ZIP...



MODIFICACIONES EN EL CÓDIGO DE PROGRAMA DEL IDE

Medida de Temperatura y Humedad con DHT11

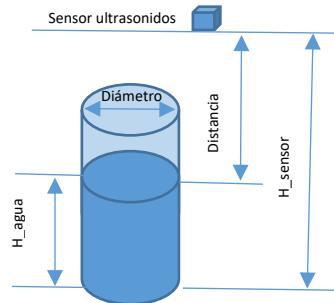
Debemos incluir en el programa las siguientes líneas de código.

1 `#include <DHT11.h> // Incluye librería de control del sensor`

2 `DHT11 dht11(7); // Define el pin 7 como pin para leer el DHT`

Cálculo de la pluviometría

A continuación, calculamos la pluviometría en función de la distancia medida por el sensor de ultrasonidos.



El volumen de la pluviometría viene dado por las fórmulas:

$$H_{agua} = Distancia * 10 - H_{sensor}$$

$$V = \frac{\pi D^2}{4} H_{agua} = 0,78 \cdot D^2 \cdot H_{agua}$$

Este volumen viene en mm^3 por lo que pasándolo a litros tendremos:

$$Pluviometría = \frac{V}{1000000}$$

La ecuación resultante será:

$$Pluviometria = 0,78 \cdot D^2 \cdot (Distancia * 10 - H_{sensor}) / 1000000$$

Por tanto, en el programa del IDE deberemos incluir la siguiente línea de comando para calcular la Pluviometría en litros:

3

```
Pluviometria=0.78 * pow(Diametro_mm,2) * (Distancia_cm*10 - H_Sensor_mm)/1000000; // Calcula el volumen en litros según la distancia medida
```

Para que la pluviometría se muestre en cm^3 la fórmula sería la misma diviendo por 1000. Es decir:

```
Pluviometria=0.78 * pow(Diametro mm,2) * (Distancia cm*10 - H Sensor mm)/1000; // Calcula el volumen en cm³ según la distancia medida
```

PROGRAMA MODIFICADO:

```

#include <DHT11.h> // Incluye librería de control del sensor
#include <Wire.h>
#include <LiquidCrystal.h>

/** Global variables */
float humi;
float temp;
int Pluviometria=0;
int Diametro_mm=100;
int Distancia_cm=0;
int H_Sensor_mm=200;
int H_agua_mm=0;
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

DHT11 dht11(7); // Define el pin 7 como pin para leer el DHT

/** Function declaration */
//bqBAT
long TP_init(int trigger_pin, int echo_pin);
long Distance(int trigger_pin, int echo_pin);

void setup()
{
  humi=0;

  temp=0;

  lcd.begin(16, 2);
  lcd.clear();

  lcd.setCursor(0,0);
  lcd.print("(°C) / (%) / (L)");

  pinMode( 9 , INPUT );

  pinMode( 8 , OUTPUT );
}

void loop()
{
  Distancia_cm=Distance(8,9);

  // Calcula el volumen en litros según la distancia medida
  Pluviometria=0.78 * pow(Diametro_mm,2) * (Distancia_cm*10 - H_Sensor_mm)/1000000;

  lcd.setCursor(0,1);
  lcd.print(String(temp) + String("/") + String(humi) + String("/") + String(Pluviometria));
  delay(1000);
}

/** Function definition */
//bqBAT
long TP_init(int trigger_pin, int echo_pin)
{
  digitalWrite(trigger_pin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigger_pin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigger_pin, LOW);
  long microseconds = pulseIn(echo_pin ,HIGH);
  return microseconds;
}

long Distance(int trigger_pin, int echo_pin)
{
  long microseconds = TP_init(trigger_pin, echo_pin);
  long distance;
  distance = microseconds/29/2;
  if (distance == 0){
    distance = 999;
  }
  return distance;
}

```

RECURSOS UTILIZADOS PARA LA ELABORACIÓN DE LAS PRÁCTICAS

Para la confección de estas prácticas, se ha utilizado las Prácticas de Arduino Nivel I de José Manuel Ruiz Gutiérrez, así como diferentes enlaces web cuyos link se incluyen en el desarrollo de las prácticas.

La programación se ha realizado con Visualino y la simulación de circuitos con [Autodesk Circuits](#)