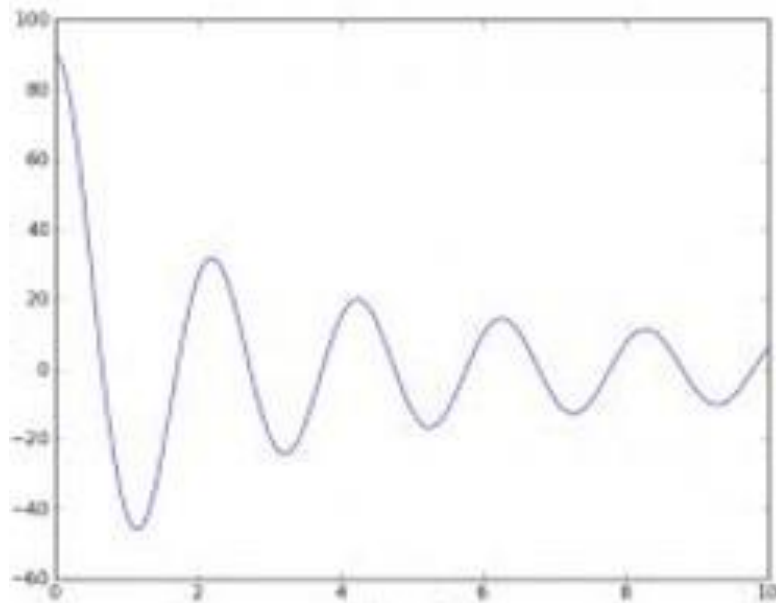


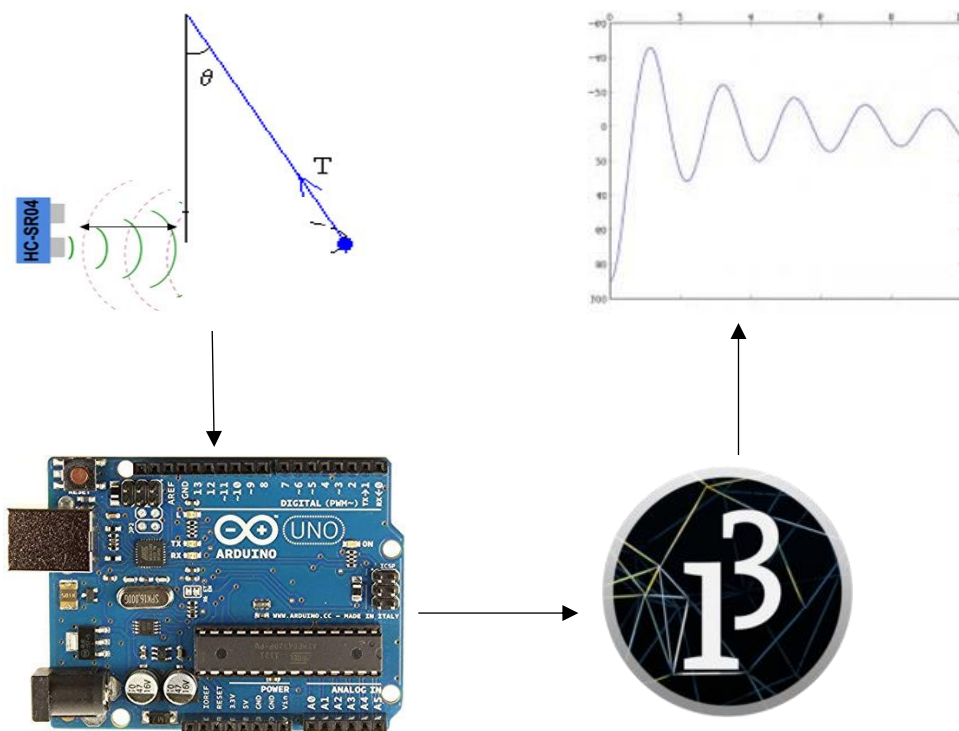
# CREAR GRÁFICA USANDO PROCESSING Y ARDUINO



NOMBRE	APELLIDO	Nº DE MATRÍCULA
DEWEI	ZHOU	54912
DAVID	MARTINEZ PRIETO	54740

## Resumen

- Vamos a tratar de representar gráficamente en la pantalla del ordenador los datos obtenidos por los sensores como: sensor de temperatura, sensor de humedad, sensor de ultrasonido, etc.
- En este caso, vamos a crear una gráfica de un péndulo simple con sensor de ultrasonido.



1º- Primero colocaremos el péndulo delante del sensor de ultrasonido a una distancia de 30cm.

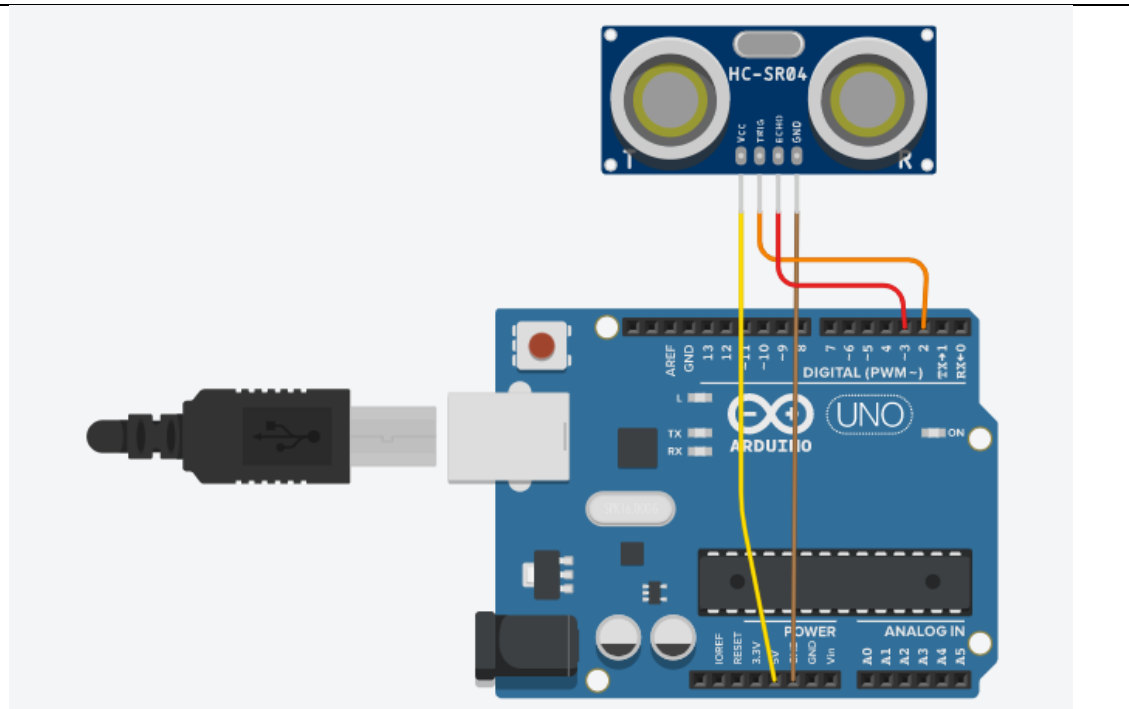
2º- Teniendo en cuenta que conocemos la longitud del péndulo y la distancia que hay entre el extremo del péndulo y la vertical o posición de equilibrio, podemos hallar el ángulo.

3º- Enviamos a **Processing** los datos obtenidos en **Arduino**.

4º- En **Processing** se creará la gráfica.

## MATERIALES

- Placa Arduino Uno R3
- Sensor de ultrasonido HC-SR04
- Péndulo



## CONECTAR EL SENSOR AL ARDUINO

Conectaremos el sensor de ultrasonido a los pines del Arduino.

- Conectaremos la **Vcc** del sensor con el pin de **5V**.
- Conectaremos la **Trig** del sensor con el pin digital **2**.
- Conectaremos la **Echo** del sensor con el pin digital **3**.
- Conectaremos la **GND** del sensor con el pin **GND**

**CÓDIGO DEL ARDUINO**

En `void setup` inicializamos la comunicación serial a 9600 bits por segundo:

```
const int trigger1 = 2; //Trigger Pin del sensor
const int echol = 3; //Echo pin del sensor
float tiempo, angulo_grado;
float dist, dist2;

void setup() {
  // inicializar la comunicación serial a 9600 bits por segundo:
  Serial.begin(9600);
  pinMode(trigger1, OUTPUT); //output (de salida)
  pinMode(echol, INPUT); // input (de entrada)
}
```

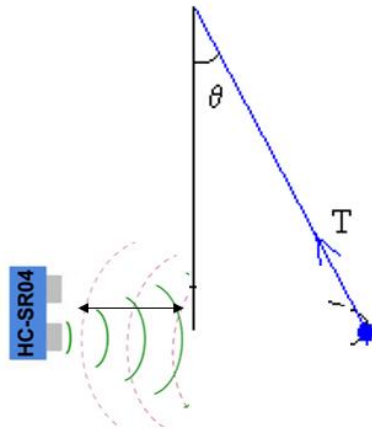
En `void loop`, mediremos el tiempo entre el envío y la recepción de un pulso sonoro y calcularemos la distancia que hay entre el sensor y el objeto.

Teniendo en cuenta que para crear la gráfica del péndulo necesitamos hallar ángulo (en grados) que forma el péndulo con la vertical o posición de equilibrio. La posición de equilibrio del péndulo estaría a 30cm del sensor, por lo tanto hay que restar treinta a la distancia obtenida para obtener 0 grados en esa posición. Si conocemos la longitud del péndulo (que en este caso es 30cm) y la distancia que hay entre el extremo del péndulo y la vertical o posición de equilibrio, podemos hallar el ángulo utilizando arcoseno.

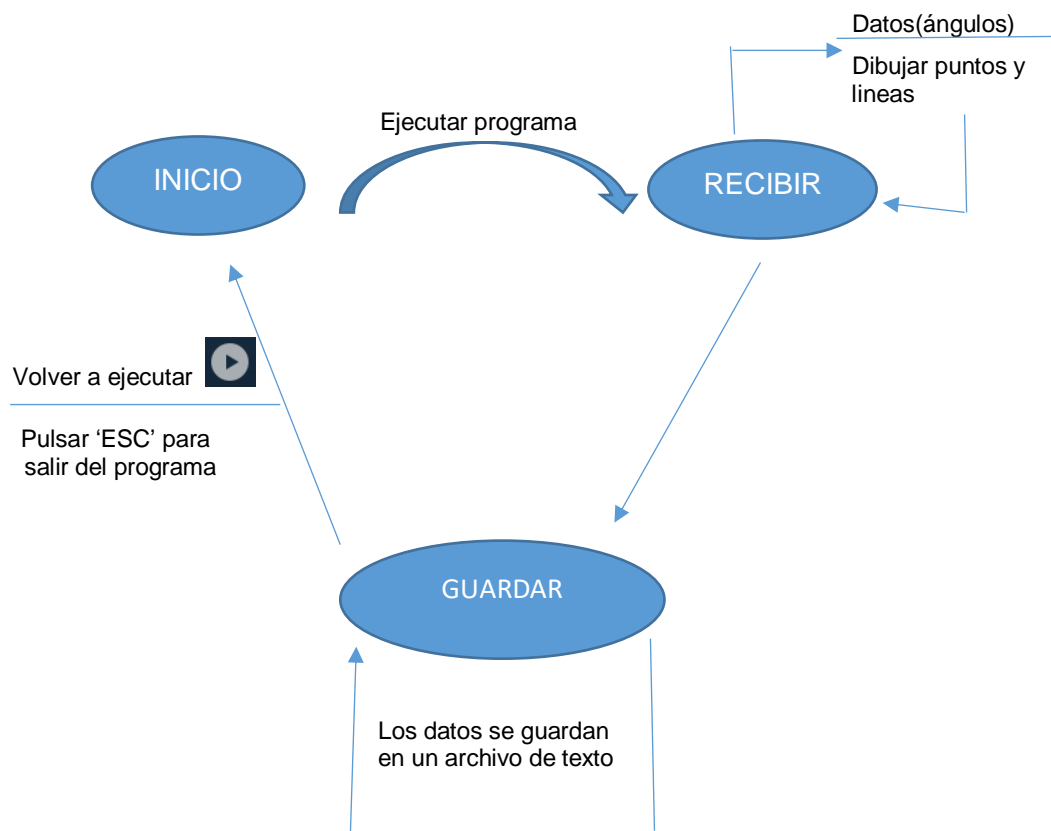
```
// la rutina de bucle se ejecuta una y otra vez para siempre:
void loop() {
  digitalWrite(trigger1, LOW); //Para generar un pulso limpio, ponemos a LOW a 2us
  delayMicroseconds(2);
  digitalWrite(trigger1, HIGH); //Genera Trigger (disparo) de 10 us
  delayMicroseconds(10);
  digitalWrite(trigger1, LOW);
  tiempo = pulseIn(echol, HIGH); //Medimos el tiempo entre pulsos, en microsegundos
  dist = tiempo * 0.034 / 2; //Calculamos y obtenemos la distancia entre el sensor y el objeto
  dist2 = dist - 30; //Situaremos el pendulo a 30cm del sensor
  //Si el pendulo esta a 30cm, obtendriamos un angulo de 0 grados
  if(dist2 < 30)
  {
    angulo_grado = asin((dist2 / 30) * 180 / 3.14); //angulo en grados

    Serial.println(angulo_grado); // Imprime los datos

    delay(50); //Retardo
  }
}
```

**PÉNDULO****ARDUINO -> PROCESSING**

En **Processing** se obtendrá los datos procedentes del **Arduino**.

**MÁQUINA DE ESTADO**

**CONDIGO EN PROCESSING**

En esta parte del código importamos la librería Serial, declaramos las variables y nombramos el puerto serie como **puertoArduino**. El **PrintWriter** se usa para crear archivos.

```
1
2  import processing.serial.*; //Importamos la librería Serial
3
4  Serial puertoArduino; //Nombre del puerto serie
5
6  //Declaracion de variables
7  int x = 65, ancho = 700, alto = 600;
8  boolean p = true; //Boolean: Nos permiten representar valores de tipo lógico (TRUE/FALSE)
9  int cFondo = 240; //Color fondo
10
11  PrintWriter datos; //Para crear el archivo de texto donde guardar los datos
12
13  // Declarar y construir un objeto (llamado g) de la class(clase) Graf
14  //Una class(clase) es un compuesto de campos (datos) y métodos (funciones que forman parte de la class) que pueden crearse como objetos.
15  //La primera letra del nombre de una class suele estar en mayúsculas para separarla de otros tipos de variables.
16  //En este caso el nombre de la class seria "Graf"
17  Graf g = new Graf(ancho, alto, cFondo);
18
```

En **void setup** crearemos una ventana de X píxeles por Y píxeles. En esta ventana crearemos nuestro entorno gráfico. También abriremos el puerto serie '**COM3**' y crearemos un archivo de texto llamado '**angulos.txt**'.

```
19
20  void setup (){
21    //Con esta función, Processing crea una ventana de X píxeles por Y píxeles. En esta ventana crearemos nuestro entorno gráfico.
22    size(700, 600);
23
24    background(240); //Determina el color del fondo de la ventana creada
25
26    //Abrir el puerto serie COM3
27    puertoArduino= new Serial(this, "COM3", 9600); // Aqui ponemos la COM del arduino que en nuestro caso es 3
28    puertoArduino.bufferUntil('\n');
29
30    // Guardaremos los datos muestreados en un archivo de texto
31    datos = createWriter("angulos.txt");
32
33    //Con "fill" indica el color de lo que esta bajo él
34    fill(0, 0, 255);
35
36    text("Angulo en grados: ", 20, 40); //Escribe un texto en la posición X, Y deseada.
37    text("Muestras", ancho / 2, alto - 20); //Escribe un texto en la posición X, Y deseada.
38    g.cuadricula(); //Iremos a la funcion "void cuadricula()" que esta dentro de "class Graf"
39
40  }
```

En **void draw**, guardaremos los datos en una variable y copiamos los datos en el archivo de texto que habíamos creado.

```
42 void draw()
43 {
44     //String es una cadena que almacena una gran variedad de caracteres.
45     String inString = puertoArduino.readStringUntil('\n'); //Lee el dato y lo almacena en la variable "inString"
46
47     if (inString != null) //si inString distinto de null
48     {
49         //trim elimina los caracteres de espacio en blanco del principio y final de una cadena.
50         inString = trim(inString);
51         float val = float(inString);
52         datos.print(val + " s    "); // copia el dato en medidas.txt
53         g.puntos(x, val, p); //Iremos a la funcion "void puntos()" que esta dentro de "class Graf"
54         p = false;
55         x = x + 20; //va sumando de 20 en 20
56         if (x > ancho - 60) //si la gráfica supera el ancho de la ventana, lo borraremos y crearemos una nueva
57         {
58             x = 60;
59             g.borra(); //Iremos a la funcion "void borra()" que esta dentro de "class Graf"
60             g.cuadricula(); //Iremos a la funcion "void cuadricula()" que esta dentro de "class Graf"
61             p = true;
62         }
63     }
64 }
```

Para crear la gráfica utilizaremos varias funciones que estaría dentro de una **class** que lo llamaremos **Graf**:

- **void** cuadricula() : Lo utilizaremos para crear líneas horizontales y verticales de la gráfica.
- **void** borra() : Lo utilizaremos para borrar la gráfica una vez que esté lleno para crea otra gráfica nueva.
- **void** puntos(int x, float nValor, boolean primera): Lo utilizaremos para dibujar líneas y puntos para crear la gráfica de un péndulo

```
74
75 class Graf
76 {
77     //declaración de variables
78     int nX, nY, colF;
79     float coordAntX, coordAntY;
80
81     Graf (int x, int y, int cF)
82     {
83         nX = x; // se almacena el valor en la variable "nX", en el que tendria 700
84         nY = y; // se almacena el valor en la variable "nY", en el que tendria 600
85         colF = cF; // se almacena el valor en la variable "colF", en el que tendria 240
86     }
87
88     void cuadrricula()
89     {
90
91         stroke(0); //permite cambiar el color de trazos o líneas, después de stroke tendrá el color de este.
92
93         for (int j = 60 ; j <= nX - 60; j = j + 20)
94         {
95             //Crear líneas. Los dos primeros valores son la primera coordenada X, Y. Los dos últimos valores son la última coordenada X, Y.
96             line (j, 60, j, nY - 60);
97         } // Creamos líneas verticales
98         for (int j = 60 ; j <= nY - 60; j = j + 20)
99         {
100             line (60, j, nX - 60, j);
101         } //Creamos líneas horizontales
102
103     }
104
105     void borra()
106     {
107         //Creamos color de fondo otra vez para poder borra las cosas que hemos creado antes
108         fill(colF); // Color del fondo
109
110         noStroke(); //Desactiva o borra los trazos (contornos)
111
112         //rect() : Los dos primeros valores es la posición X, Y en el gráfico. Los dos últimos valores son la anchura y la altura
113
114         //Modifica la ubicación desde la cual se dibujan los rectángulos cambiando la forma en
115         // que se interpretan los parámetros dados a rect ().
116         rectMode(CORNERS);
117         //rectMode (CORNERS) interpreta los dos primeros parámetros de rect () como la ubicación de una esquina,
118         //y los parámetros tercero y cuarto como la ubicación de la esquina opuesta.
119         rect(50 , 50, nX , nY - 30 );
120     }
121
```



```

122 void puntos(int x, float nValor, boolean primera)
123 {
124
125     fill(255,255,255);//color de fondo
126     rectMode(CORNERS);
127     //rectMode (CORNERS) interpreta los dos primeros parámetros de rect () como la ubicación de una esquina,
128     //y los parámetros tercero y cuarto como la ubicación de la esquina opuesta.
129     rect(140,25,200,45);//Los dos primeros valores es la posición X, Y en el gráfico. Los dos últimos valores son la anchura y la altura
130     fill (0,0,255);
131     text(nValor, 142, 40); //Escribe el valor de la variable "nValor" en la posición X, Y deseada.
132     fill(0, 0, 255);
133     float v = map(5*nValor, 0, 1023, nY - 60, 60); //Mapeo inverso entre
134                                     //los márgenes sup e inf.
135     ellipse(x, v-250, 5, 5);//Creamos puntos
136
137     //Une los dos puntos con una línea excepto en la primera lectura.
138     if (primera == false)
139     {
140         line (coordAntX, coordAntY-250, x, v-250);
141
142     }
143     //guardamos las cordenadas del ultimo punto para poder crear una línea que una los puntos
144     coordAntX = x;
145     coordAntY = v;
146 }
147
148 //El resultado saldría una línea continua
149 }
150 //Con el mismo código podemos crear gráfica con los datos obtenidos de otros sensores como sensor de temperatura o de humedad.

```

En la función **void keyPressed()** guardaremos los datos en el archivo y saldremos del programa pulsando la tecla **'ESC'**.

```

66 void keyPressed() //Presionar 'ESC' para salir
67 {
68     datos.flush(); // Escribe los datos en el archivo
69     datos.close(); // Final del archivo
70     exit(); //Salimos del programa
71 }

```

Con el mismo código de **Processing** podemos crear gráficas utilizando otros sensores como sensor de temperatura, sensor de humedad.etc