

## **Agenda**

## **Índice**

<b>1. Presentación</b>	<b>1</b>
1.1. ¿Quienes somos? . . . . .	1
1.2. Requisitos . . . . .	3
<b>2. Arduino</b>	<b>3</b>
2.1. Intro . . . . .	3
2.2. Montaje . . . . .	3
2.3. Conceptos CPP . . . . .	5
<b>3. SODAR</b>	<b>6</b>
3.1. Movimiento . . . . .	6
3.2. Sensor . . . . .	7
<b>4. Processing</b>	<b>9</b>
4.1. Introducción . . . . .	9
4.2. Geometría . . . . .	11

## **1. Presentación**

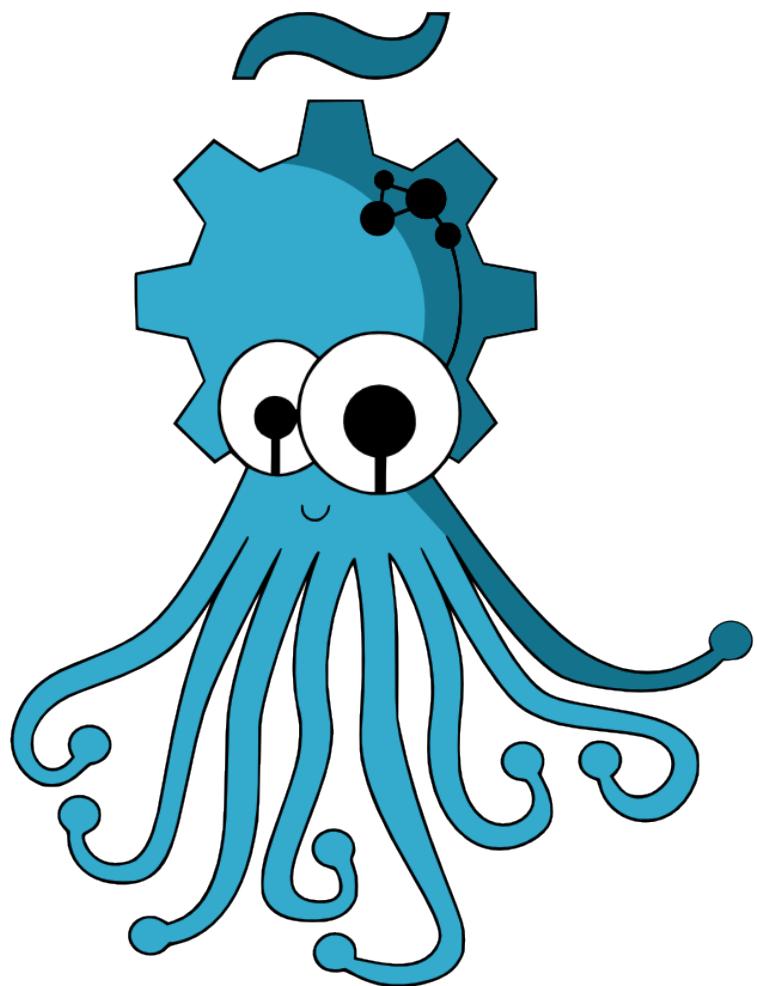
### **1.1. ¿Quienes somos?**

BricoLabs y la OSHWDem



# BricoLabs

<http://bricolabs.cc/>



<http://oshwdem.org>

### PONENTES

- @ctemes
- @pepdiz
- @Milo\_1008

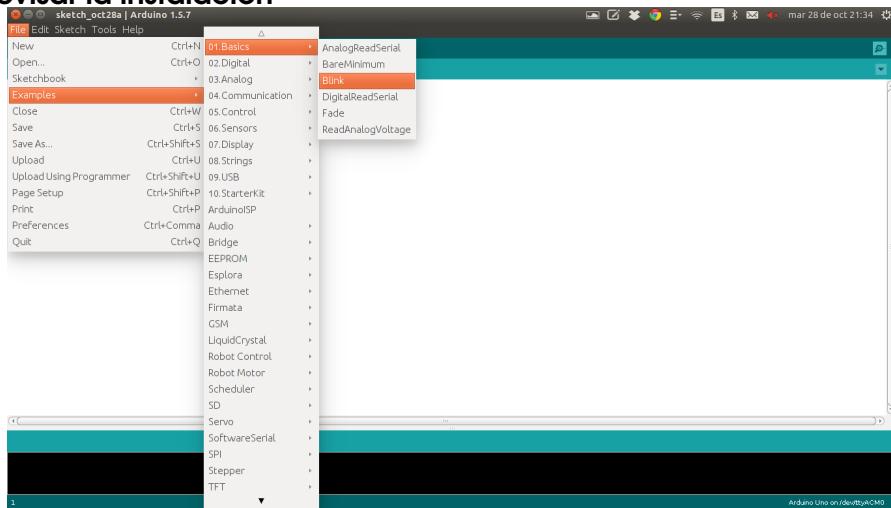
- @salvari

## Asistentes

- ¿Quién ha programado antes?
- ¿Quién conoce el Arduino?
- ¿Quién conoce Processing?
- ¿Traéis los deberes hechos? ;)

## 1.2. Requisitos

### Revisar la instalación



## 2. Arduino

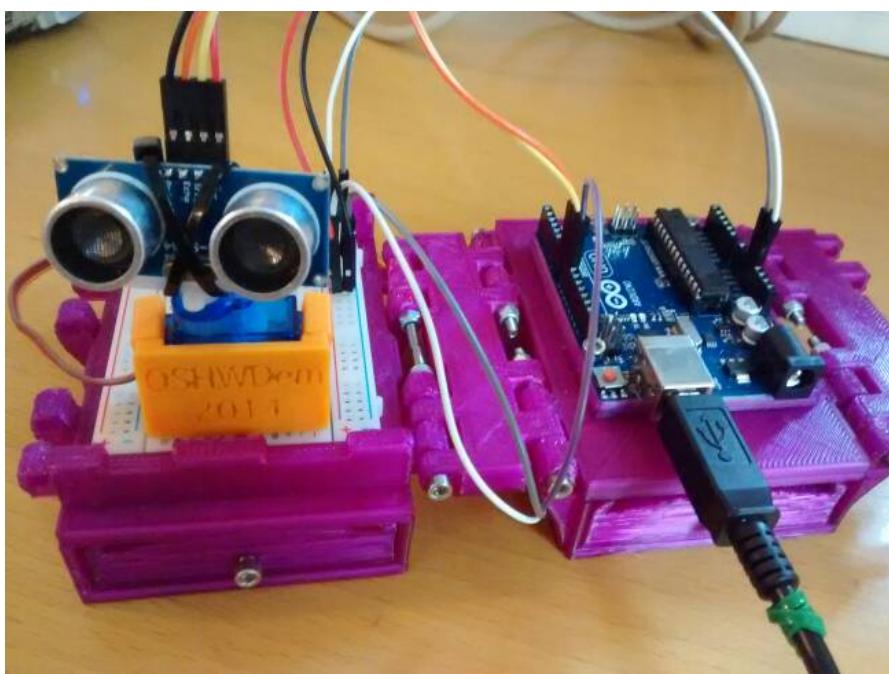
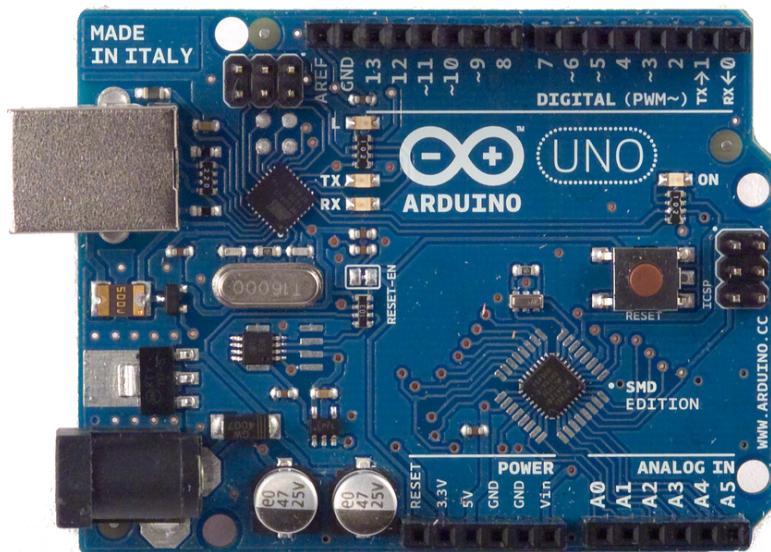
### 2.1. Intro

#### Arduino

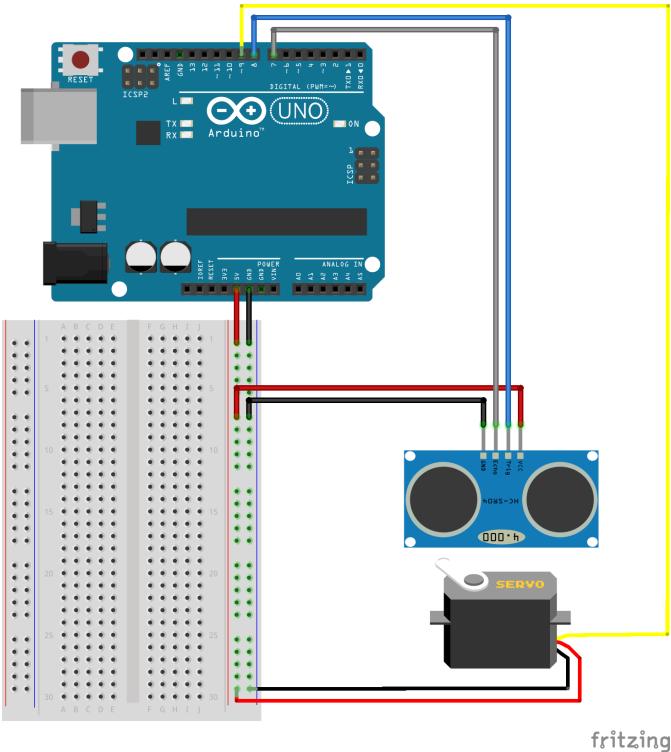
Página Principal Foto Familia

### 2.2. Montaje

#### Montaje I



## Montaje II



fritzing

## 2.3. Conceptos CPP

### Estructura de un programa Arduino

```
#include <Servo.h>

#define SERVO_PWM_PIN 9

Servo myservo;

/*-----*
 *setup
 * Se ejecuta una sola vez al principio del programa. O cuando el arduino
 * se resetea.
 *-----*/
void setup() {
}

/*-----*
 *loop
 * Se ejecuta siempre, hasta el fin de los tiempos :-)
 *-----*/
void loop() {
}
```

### Función

```
int medida(int intentos) {
}

void canta(){
    Serial.println("La, la, la!");
}
```

## Serial

```
Serial.begin(9600);
Serial.print("Hola_"); // Sin cambiar de linea
Serial.println("mundo"); // Con retorno de linea
Serial.print(10, DEC); // Con formato (DEC, BIN, OCT, HEX)
Serial.println(10.1234, 2); // Imprime 10.12
```

## 3. SODAR

### 3.1. Movimiento

#### Servo



```
#include <Servo.h>
#define SERVO_PWM_PIN 9
Servo myservo;
myservo.attach(SERVO_PWM_PIN);
myservo.write(angle);
```

#### Barridos



## Una solución

```
#define ANGULO_MIN 0      // angulo minimo del barrido (en grados)
#define ANGULO_MAX 180     // angulo maximo del barrido (en grados)
#define PASO_ANGULO 1       // paso de barrido

#define ATRAS -1
#define ADELANTE 1

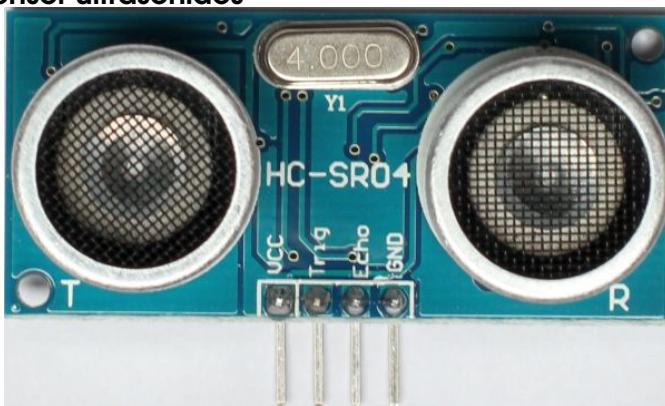
loop(){
    delay(50);           // espera 50 milisegundos
    miservo.write(angulo); // avanza el servomotor al angulo indicado

    if (angulo >= ANGULO_MAX) { direccion = ATRAS; }
    if (angulo <= ANGULO_MIN) { direccion = ADELANTE; }

    angulo += direccion * PASO_ANGULO; // incrementa el angulo de
                                      // barrido un paso de barrido
}
```

### 3.2. Sensor

#### Sensor ultrasonidos



## Electric Parameter

<b>Working Voltage</b>	DC 5 V
<b>Working Current</b>	15mA
<b>Working Frequency</b>	40Hz
<b>Max Range</b>	4m
<b>Min Range</b>	2cm
<b>MeasuringAngle</b>	15 degree
<b>Trigger Input Signal</b>	10uS TTL pulse
<b>Echo Output Signal</b>	Input TTL lever signal and the range proportion
<b>Dimension</b>	45*20*15mm

Sensor de distancia HC-SR04

### Protocolo

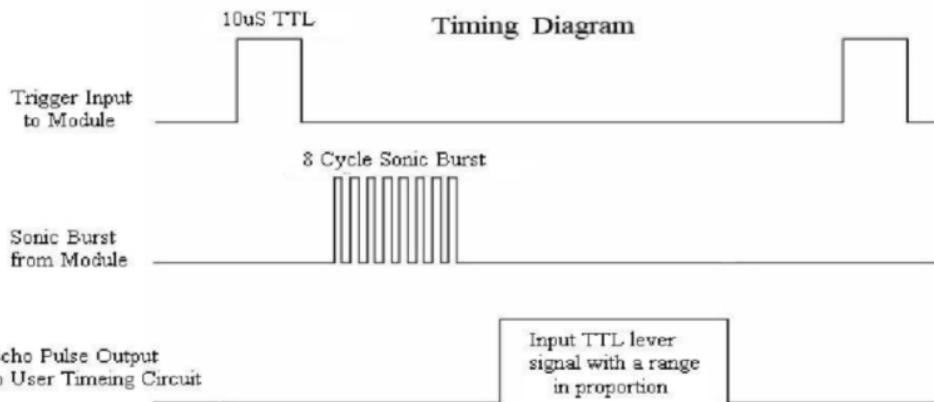


Diagrama de señales

### Función Medida Una función de bajo nivel

```
/*
 *-----*
 * setup
 * Se ejecuta una sola vez al principio del programa. O cuando el arduino
 * se resetea.
 *-----*/
void setup() {
    pinMode(TRIGGER_PIN, OUTPUT); // pin trigger es salida
    pinMode(ECHO_PIN , INPUT ); // pin echo es entrada
    Serial.begin(9600); // Abrimos el puerto serie
}

/*
 *-----*
 * loop
 * Se ejecuta siempre repetidamente, hasta el fin de los tiempos :-)
 *-----*/
void loop() {
    // Preparamos el sensor
    digitalWrite(TRIGGER_PIN, LOW); // Nivel bajo para estabilizar
    delayMicroseconds(5); // garantizamos 5 microsegundos

    digitalWrite(TRIGGER_PIN, HIGH); // Enviamos un pulso de 10 microsegundos
    delayMicroseconds(10);
    digitalWrite(TRIGGER_PIN, LOW);
}
```

```

time = pulseIn(ECHO_PIN, HIGH);           // Leemos el echo, viene codificado en
                                         // el ancho del pulso
dist = time / 29 / 2;                   // Calculamos la distancia
                                         // y la imprimimos
Serial.println(dist);
delay(60);
}

```

## NewPing

```

#include <NewPing.h>

NewPing sonar(TRIGGER_PIN,
              ECHO_PIN,
              DISTANCIA_MAXIMA); // Creamos un objeto sensor

int cm = sonar.ping_cm();             // Medimos

```

## Solución SODAR

```

-----*
obtenerDistanciaEnviar
esta funcion obtiene la distancia a la que se encuentra un objeto
y envia por el puerto serie el angulo y la distancia del objeto encontrado
-----*/

int obtenerDistanciaEnviar(int angulo) {

    int cm = sonar.ping_cm();      // obtiene la distancia en cm
    Serial.print(angulo, DEC);     // envia por puerto serie la distancia y el angulo
    Serial.print(",");
    Serial.println(cm, DEC);       // formato:
                                    // angulo,distancia<FINLINEA>
}

#define ANGULO_MIN 0    // angulo minimo del barrido (en grados)
#define ANGULO_MAX 180 // angulo maximo del barrido (en grados)
#define PASO_ANGULO 1   // paso de barrido

#define ATRAS -1
#define ADELANTE 1

loop(){
    delay(50);                  // espera 50 milisegundos
    miservo.write(angulo);       // avanza el servomotor al angulo indicado

    obtenerDistanciaEnviar(angulo); // obtiene la distancia y envia
                                    // los datos por el puerto serie

    if (angulo >= ANGULO_MAX) { direccion = ATRAS; }
    if (angulo <= ANGULO_MIN) { direccion = ADELANTE; }

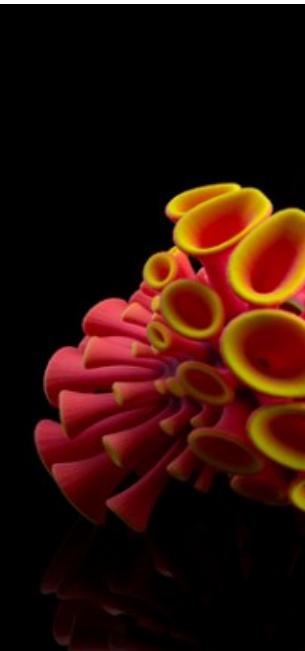
    angulo += direccion * PASO_ANGULO; // incrementa el angulo de
                                      // barrido un paso de barrido
}

```

## 4. Processing

### 4.1. Introducción

#### Processing



## Estructura de un programa

```
import processing.serial.*;  
  
// ----- variables -----  
  
int numPuntos=0;  
String contenidoSerie;  
Serial miPuerto;  
  
/*-----  
setup  
Se ejecuta una sola vez al principio del programa  
-----*/
```

```
void setup() {
    background(255);
    size(MAXX, MAXY);
}

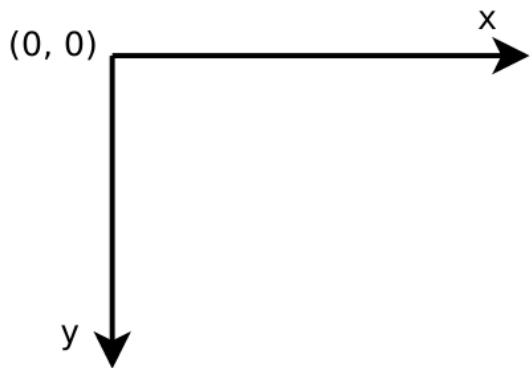
-----
draw
Se ejecuta eternamente hasta el fin de los tiempos
-----
void draw() {

}
```

---

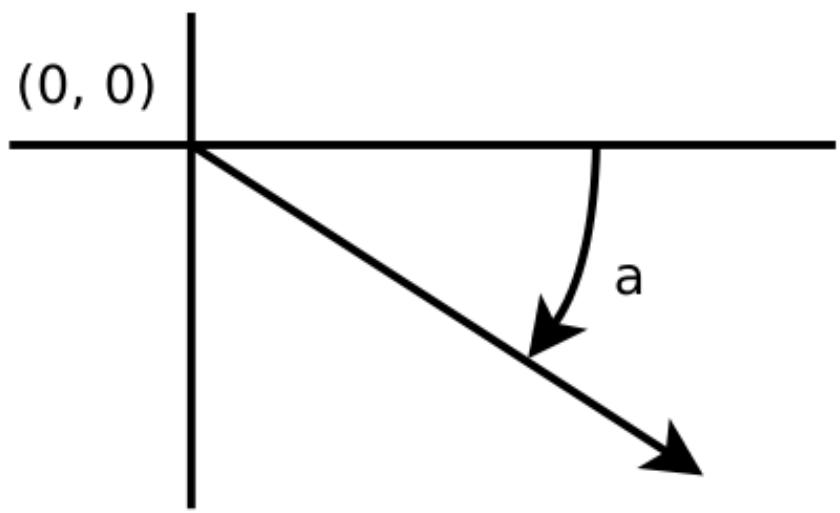
## 4.2. Geometría

### Pantalla de Processing Coordenadas



Coordenadas en Processing

### Pantalla de Processing Ángulos



$$\beta = 2\pi - \text{radians}(\alpha)$$