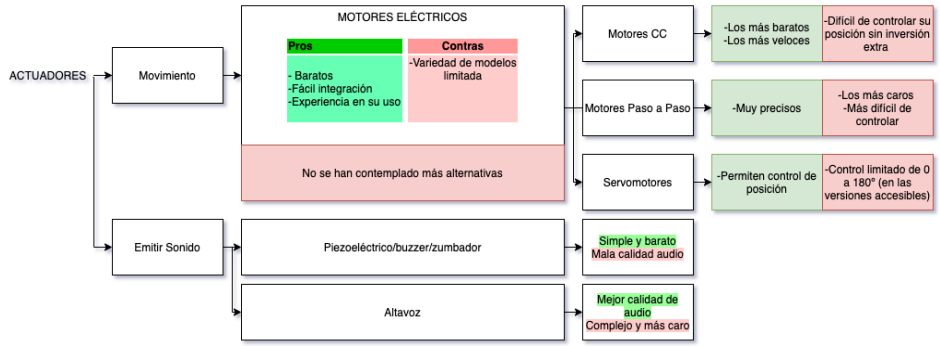
[Datos de salida]

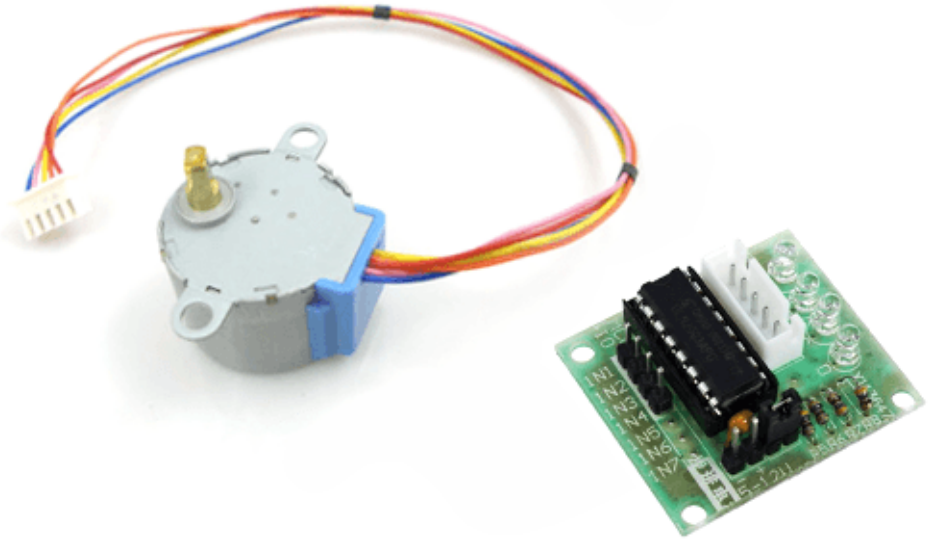
Nuestro proyecto necesita de varios actuadores para llevarse a cabo, nuestras necesidades son de movimiento y de sonido.
Por ello planteamos lo siguiente:



A partir de esto concluimos que lo mejor es:

- Para el movimiento de los ejes del plotter motores paso a paso, para subir y bajar el rotulador un servo.
- Para emitir los sonidos el buzzer.

Motor paso a paso
Los más fáciles de encontrar, baratos y con mucha documentación en su uso con Arduino son los 28Byj-48.
fuente imagen y código: *Luis llamas*



En el enlace se detalla bien el funcionamiento, a modo de resumen, para controlar este motor se encienden de forma secuencial unas partes o otros del bobinado del motor, lo que permite un giro controlado y al conocer el numero de veces que se aplica la secuencia se puede conocer la

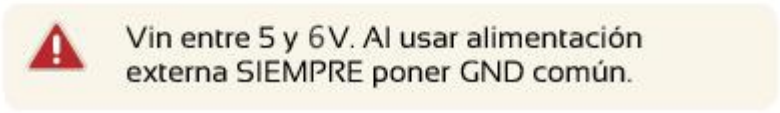
posición del motor.Se controla por lo tanto desde 4 pines de la placa, conectados al controlador del motor, el cuál necesita también 2 terminales para la corriente, funciona con 5v.

```
//Este ejemplo de codigo define las secuencias de control en ambos sentidos y realiza 2 vueltas en cada sentido
//definicion de pins
const int motorPin1 = 8;     // 28BYJ48 In1
const int motorPin2 = 9;     // 28BYJ48 In2
const int motorPin3 = 10;    // 28BYJ48 In3
const int motorPin4 = 11;    // 28BYJ48 In4

//definicion variables
int motorSpeed = 1200;   //variable para fijar la velocidad
int stepCounter = 0;     // contador para los pasos
int stepsPerRev = 4076;  // pasos para una vuelta completa

//tablas con la secuencia de encendido (descomentar la que necesiteis)
//secuencia 1-fase
//const int numSteps = 4;
//const int stepsLookup[4] = { B1000, B0100, B0010, B0001 };

//secuencia 2-fases
//const int numSteps = 4;
//const int stepsLookup[4] = { B1100, B0110, B0011, B1001 };

//secuencia media fase
const int numSteps = 8;
const int stepsLookup[8] = { B1000, B1100, B0100, B0110, B0010, B0011, B0001, B1001 };


void setup()
{
  //declarar pines como salida
  pinMode(motorPin1, OUTPUT);
  pinMode(motorPin2, OUTPUT);
  pinMode(motorPin3, OUTPUT);
  pinMode(motorPin4, OUTPUT);
}

void loop()
{
  for (int i = 0; i < stepsPerRev * 2; i++)
  {
    clockwise();
    delayMicroseconds(motorSpeed);
  }
  for (int i = 0; i < stepsPerRev * 2; i++)
  {
    anticlockwise();
    delayMicroseconds(motorSpeed);
  }
  delay(1000);
}

void clockwise()
{
  stepCounter++;
  if (stepCounter >= numSteps) stepCounter = 0;
  setOutput(stepCounter);
}

void anticlockwise()
{
  stepCounter--;
  if (stepCounter < 0) stepCounter = numSteps - 1;
  setOutput(stepCounter);
}

void setOutput(int step)
{
  digitalWrite(motorPin1, bitRead(stepsLookup[step], 0));
  digitalWrite(motorPin2, bitRead(stepsLookup[step], 1));
  digitalWrite(motorPin3, bitRead(stepsLookup[step], 2));
  digitalWrite(motorPin4, bitRead(stepsLookup[step], 3));
}
```

a

Aunque este ejemplo muestra muy bien el funcionamiento del motor, se ha encontrado una libreria que simplifica el control, se trata de *Tiny Stepper*,en su documentación puede encontrarse ejemplos de implementación.

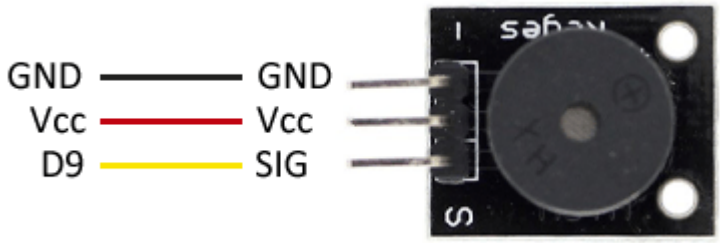
Por otra parte para subir y bajar el rotulador utilizaremos un servomotor, uno de alta disponibilidad y ampliamente utilizado con Arduino es el Sg90, que aunque sus características son reducidas, sirve de sobra para nuestras necesidades.
Fuente código e imagen: *Luis llamas* Este servomotor se controla mediante 1 pin de la placa que debe poder emitir PWM, señal de pulso modulado, es con la variación de duración de la señal como la placa interna del servo lo coloca en la posición que le damos.Una forma de identificar estos pines en la placa es porque junto al número tienen un ~.
Recibe 5v.Una limitación es que solo puede moverse en 180º.



Arduino posee una librería, incluida en el IDE para controlar este tipo de motores, la Servo.h

```
//Este ejemplo mueve alternativamente el servo de 0 a 180 y de 180 a 0 grados
#include <Servo.h>

Servo myservo;  // crea el objeto servo

int pos = 0;    // posicion del servo

void setup() {
   myservo.attach(9);  // vincula el servo al pin digital 9
}

void loop() {
   //varia la posicion de 0 a 180, con esperas de 15ms
   for (pos = 0; pos <= 180; pos += 1)
   {
      myservo.write(pos);
      delay(15);
   }

   //varia la posicion de 0 a 180, con esperas de 15ms
   for (pos = 180; pos >= 0; pos -= 1)
   {
      myservo.write(pos);
      delay(15);
   }
}
```

Por último también queremos emitir sonidos, decidimos usar un buzzer por su simplicidad, se trata de un actuador que recibe una señal eléctrica y mediante un oscilador interior genera un pitido de un tono determinado, suficiente para nuestra aplicación.
Fuente imagen y código: *Luis llamas*




```
// El siguiente código emite 2 tonos alternativamente
const int pinBuzzer = 9;

void setup()
{
}

void loop()
{
  //generar tono de 440Hz durante 1000 ms
  tone(pinBuzzer, 440);
  delay(1000);

  //detener tono durante 500ms
  noTone(pinBuzzer);
  delay(500);

  //generar tono de 523Hz durante 500ms, y detenerlo durante 500ms.
  tone(pinBuzzer, 523, 300);
  delay(500);
}
```