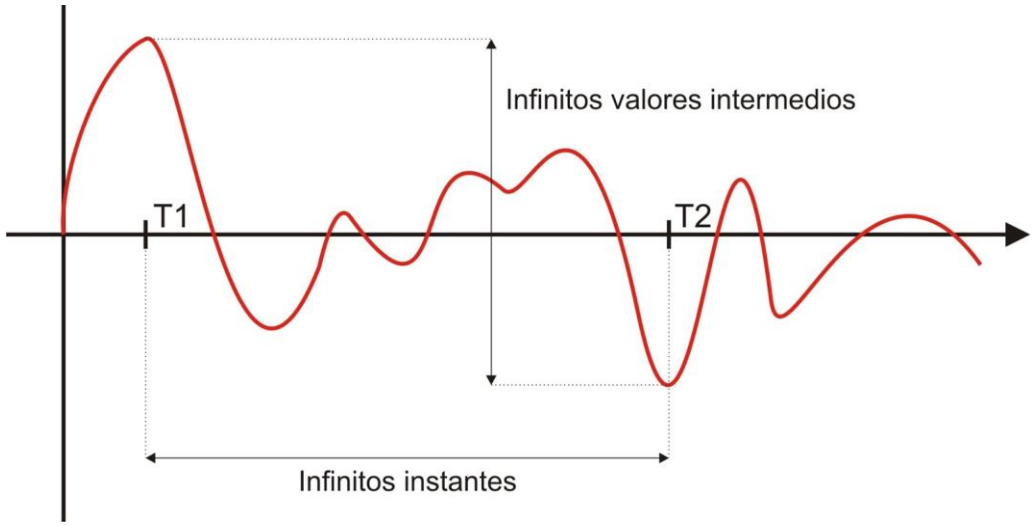
Conversión Análoga DIGITAL

Introducción

Consiste en la transcripción de señales analógicas en señal digital, con el propósito de facilitar su procesamiento (codificación, comprensión, etcétera) y hacer la señal resultante (digital) mas inmune al ruido y otras interferencias a las que son mas sensibles las señales analógicas. (Serbal, 2009)

1SEÑAL ANALOGICA

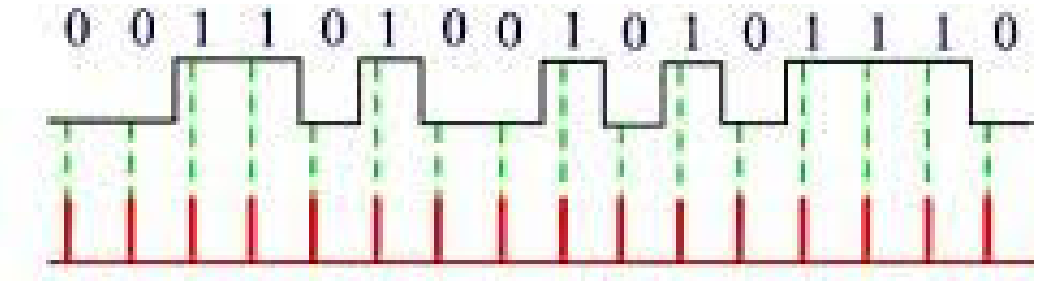
En la naturaleza, el conjunto de señales que percibimos es analógicas, así la luz, el sonido, la energía etc., son señales que tienen una variación continua. Incluso la descomposición de la luz en el arco iris vemos como se realiza de una forma suave y continua.



SEÑAL DIGITAL

Se trata de la señal cuyos signos representan ciertos valores discretos que contienen información codificada. Los sistemas que emplean señales digitales suelen apelar a la lógica binaria, de dos estados, los cuales son remplazados por unos y ceros, que indican el estado alto o bajo del nivel de tensión eléctrica.

Una señal digital pierde poca calidad y puede reconstruirse por un proceso de regeneración. Estas señales, además, pueden procesarse de manera sencilla y son poco susceptibles al ruido ambiental.



CONVERSIÓN ANALOGICA DIGITAL

Para realizar esa tarea, el conversor ADC (Analog-to-Digital Converter-Conversor Analógico Digital) tiene que efectuar los siguientes procesos:

Muestreo de la señal analógica.

Cuantización de la propia señal

Codificación del resultado de la cuantización, en código binario.

CONVERSIÓN ANALOGICA DIGITAL

1.4.1. Muestreo

Para convertir una señal analógica en digital el primer paso consiste en realizar un muestreo (sampling) de esta, o lo que es igual, tomar diferentes muestras de tensiones o voltajes en diferentes puntos de la onda senoidal. La frecuencia a la que se realiza el muestreo se denomina razón, taza o también frecuencia de muestreo y se mide en kilohertz (kHz). En el caso de una grabación digital de audio, a mayor cantidad de muestras tomadas, mayor calidad y fidelidad tendrá la señal digital resultante.

Durante el proceso de muestreo se asignan valores numéricos equivalentes a la tensión o voltaje existente en diferentes puntos de la sinusoide, con la finalidad de realizar a continuación (EcuRed, 2005).

CONVERSIÓN ANALOGICA DIGITAL

Las tasas o frecuencias de muestreo mas utilizadas para audio digital son las siguientes:

24 000 muestras por segundo (24 kHz)

30 000 muestras por segundo (30 kHz)

44 100 muestras por segundo (44,1 kHz) (Calidad de CD)

48 000 muestras por segundo (48 kHz)

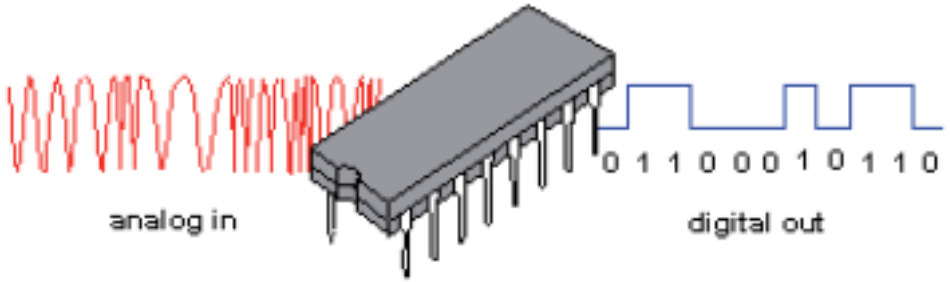
CONVERSIÓN ANALOGICA DIGITAL

La siguiente tabla muestra los valore numéricos del 0 al 7, pertenecientes al sistema decimal y sus equivalentes en código numérico binario. En esta tabla se puede observar que utilizando solo tres bits por cada numero en código binario, se puede representar ocho niveles o estados de cuantización (Serbal, 2009)

Valores en volt Sistema Decimal	Conversión a Código Binario
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

1CONVERSOR ANALOGO DIGITAL EN ARDUINO

Un conversor analógico-digital en un dispositivo electrónico capaz de convertir una señal analógica en un valor binario, en otras palabras, este se encarga de transformar señales analógicas a digitales (0 y 1).



Un dispositivo establece una relación entre su entrada (señal analógica) y su salida (digital) dependiendo de su resolución. La resolución determina la precisión con la que se reproduce la señal original.

CONVERSOR ANALOGO DIGITAL EN ARDUINO

Esta resolución se puede saber, siempre y cuando conozcamos el valor de la entrada a convertir y la cantidad máxima de la salida en dígitos binarios.

Resolucion = +Vref/2^n (donde n son bits)

La tarjeta Arduino utiliza un conversor A/D de 10-bits, así que: Resolución = Vref/1024 Mapeara los valores de voltaje de entrada, entre 0 y Vref voltios, a valores enteros comprendidos entre 0 y 1023(2^n-1). Con esas palabras, esto quiere decir que nuestros sensores analógicos están caracterizados con un valor predeterminado entre 0 y 1023.

Si Vref es igual a 5v, la resolución es aproximadamente de 5 milivoltios. Por lo tanto el error en las medidas de voltaje será siempre 5 milivoltios. (Serbal, 2009).

EJERCICIOS DE APLICACIÓN

Programación en Arduino

Esta función nos inicia la comunicación serie. Por lo tanto, no podemos enviar ni recibir ningún dato a través de la comunicación serie si no hemos ejecutado esta función. A la función hay que pasarle como parámetro la velocidad. En nuestro caso elegiremos 9600 bits por segundo: (EcuRed, 2005)

```
Void setup(){

Serial.begin(9600);}
```

A continuación, escribiremos el contenido de la función loop (la función que se ejecuta continuamente).

```
void loop() {
```

EJERCICIOS DE APLICACIÓN

Conversor análogo – digital

Programación en Arduino

En primer lugar necesitamos un lugar en el que almacenar los datos obtenidos del conversor. Este lugar se denomina variable. Para declarar una variable (es decir, para solicitar este lugar de almacenamiento) debemos indicar que tipo de dato vamos a almacenar. Sabemos que el conversor devuelve valores enteros comprendidos entre 0 y 1023 (ya que es de 10 bits). El tipo de variable que permite almacenar un dato entero es el tipo int. A la variable debemos darle un nombre para poder identificarla así que elegiremos el nombre valorADC. Así pues, el código quedara de la forma:

```
int valorADC;
```

1.6. EJERCICIOS DE APLICACIÓN

1.6.1. Conversor análogo – digital

Programación en Arduino

No obstante un valor de 0 a 1023 no es intuitivo por lo que necesitamos convertirlo a un valor de voltaje. Dado que este valor tendrá numero decimales, el valor obtenido no lo podemos almacenar en esta variable así que creamos otra que sea de tipo float:

```
float voltaje;
```

Una vez creada las variables que necesitamos procedemos a tomar la medida utilizando la función analogRead() y pasándole como argumento el pin desde el cual debe leer en conversor (en este caso el pin0 del puerto A). Almacenamos el resultado en la variable valorADC:

```
valorADC = analogRead(A0)
```

EJERCICIOS DE APLICACIÓN

Conversor análogo – digital

Programación en Arduino

A continuación convertimos este valor a un valor de voltaje sabiendo que un valor de 1023 corresponde a 5C. Por lo tanto, el valor de voltaje será el valor medido multiplicado por 5 y dividido entre 1023:

voltaje= valorADC * 5.0/1023,0;

EJERCICIOS DE APLICACIÓN

Conversor análogo – digital

Programación en Arduino

Finalmente enviamos el valor obtenido a través del puerto serie y esperamos 2m antes de realizar la siguiente medida (ya que el conversor necesita que transcurra un tiempo entre una medida y la siguientes para que los resultados sean correctos):

```
Serial.println(voltaje);

delay(2)
```

Por ultimo solo queda compilar el código y descargarlo en la placa. Una vez que este funcionando solo tenemos que abrir el monitor serie (“serial monitor”) del IDE de Arduino y ver el valor de tensión. Si vamos girando el potenciómetro comprobamos como va cambiando el valor medido.

EJERCICIOS DE APLICACIÓN

Conversor análogo – digital

Programación en Arduino

```
void setup() {
Serial.begin(9600);//inicializa la comunicacion serial
}
void loop() {
int valorADC; //valor de 0 a 1023
float voltaje; // variable para almacenar numero con decimales
valorADC = analogRead(A0); // almacenamos el resultado
voltaje = valorADC * (5.0 / 1023.0); // Convertidor el valor a un valor de volta
Serial.println(voltaje);//enviamos el valor obtenido a través del puerto serie
delay(2);//esperamos 2ms
}
```

EJERCICIOS DE APLICACIÓN

Conversor análogo-digital (Aumentar luz led)

Descripción

En este programa procedemos por medio de una conversión análogo – digital vamos a ir aumentando la velocidad de encendido de leds, este programa funcionara con un potenciómetro el cual ira regulando la velocidad de los leds incorporados

EJERCICIOS DE APLICACIÓN

Conversor análogo-digital (Aumentar luz led)

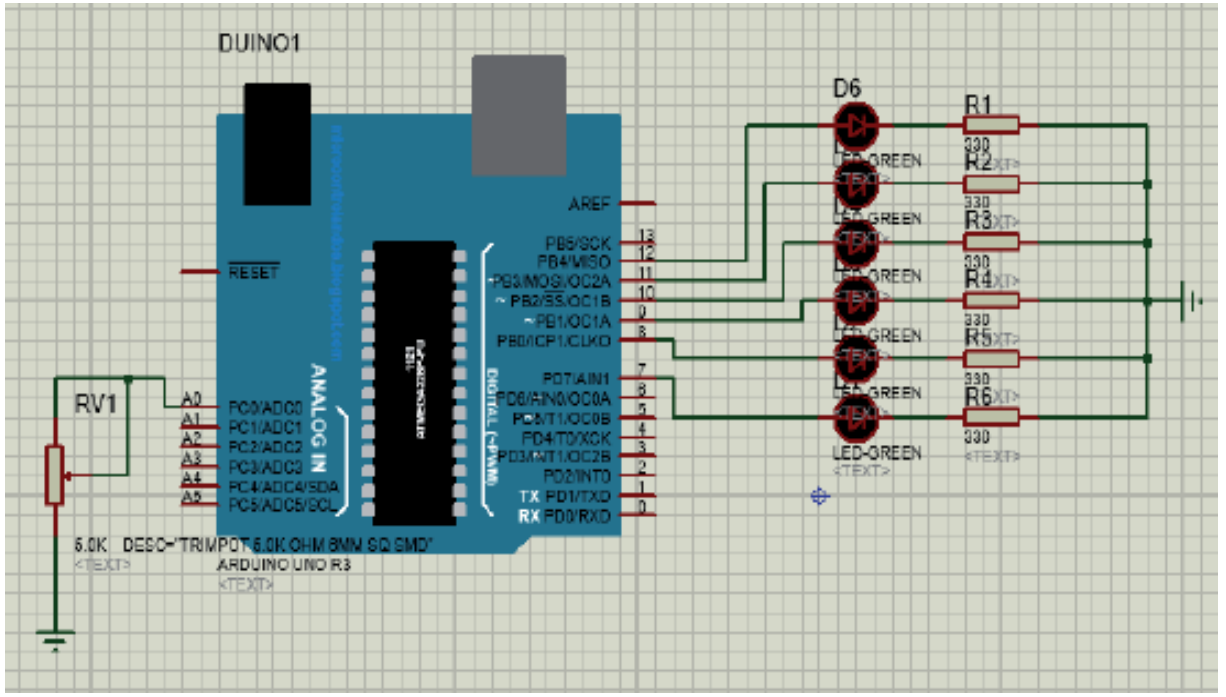
Programa En Arduino

```
void setup() {
  pinMode(led5, OUTPUT); //pines de salida
  pinMode(led6, OUTPUT); //pines de salida
  pinMode(led7, OUTPUT); //pines de salida
  pinMode(led8, OUTPUT); //pines de salida
  pinMode(led9, OUTPUT); //pines de salida
  pinMode(led10, OUTPUT); //pines de salida
}

void loop() {
  var = analogRead(0); //Aqui le decimos que lea el valor del potenciómetro, valor el cual oscila entre 0 y 1
  analogWrite(led5, var / 4);
  delay(50);//tiempo de espera
  analogWrite(led6, var / 4);
  delay(50);//tiempo de espera
  analogWrite(led7, var / 4);
  delay(50);//tiempo de espera
  analogWrite(led8, var / 4);
  delay(50);//tiempo de espera
  analogWrite(led9, var / 4);
  delay(50);//tiempo de espera
  analogWrite(led10, var / 4);
  delay(50);//tiempo de espera
}
```

EJERCICIOS DE APLICACIÓN

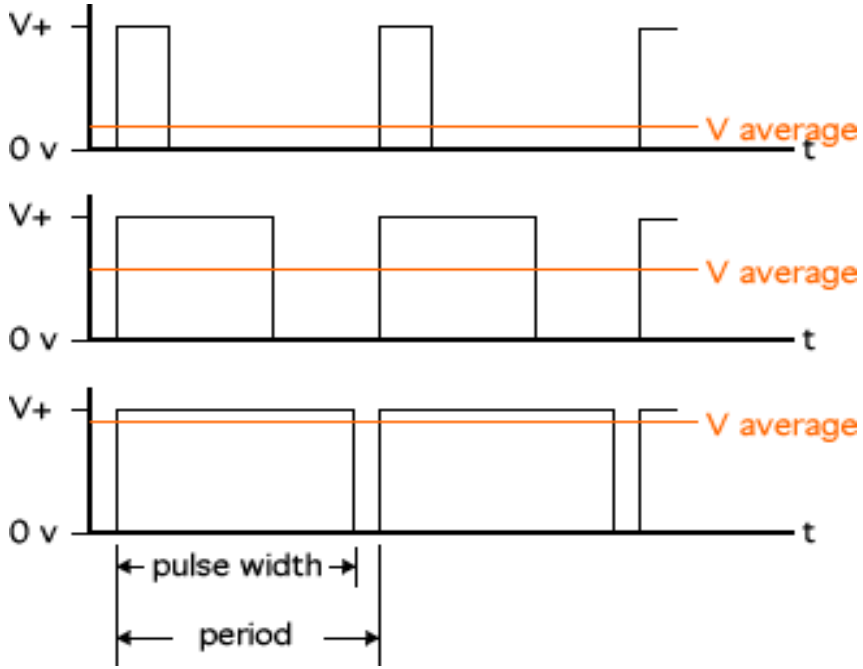
Conversor análogo-digital (Aumentar luz led)

Simulación



PWM (Control De Ancho De Pulso)

La conversión análoga digital permite traer valores análogos al Arduino con una referencia de 0 a 1023, el control de ancho de pulso permite enviar esa referencia por un pin determinado, lo que hacemos es modificar un pulso eléctrico el tiempo que se mantenga en 1 lógico.



PWM (Control De Ancho De Pulso)

Programación: El ancho de pulso esta divido en referencias de 0 a 255 y en este caso

solo podemos usar los pines digitales que posee un guion en la numeración de pines.

Lo que cambia de nuestra sección es que ahora en vez de usar un analogRead, lo cambiamos por AnalogWrite, en los pines antes mencionados.

El programa a continuación controlaremos la velocidad de un motor observando el valor

análogo en la comunicación serial mediante un potenciómetro por ende debemos al

valor análogo dividirlo para 4 para tener una referencia hasta 255.

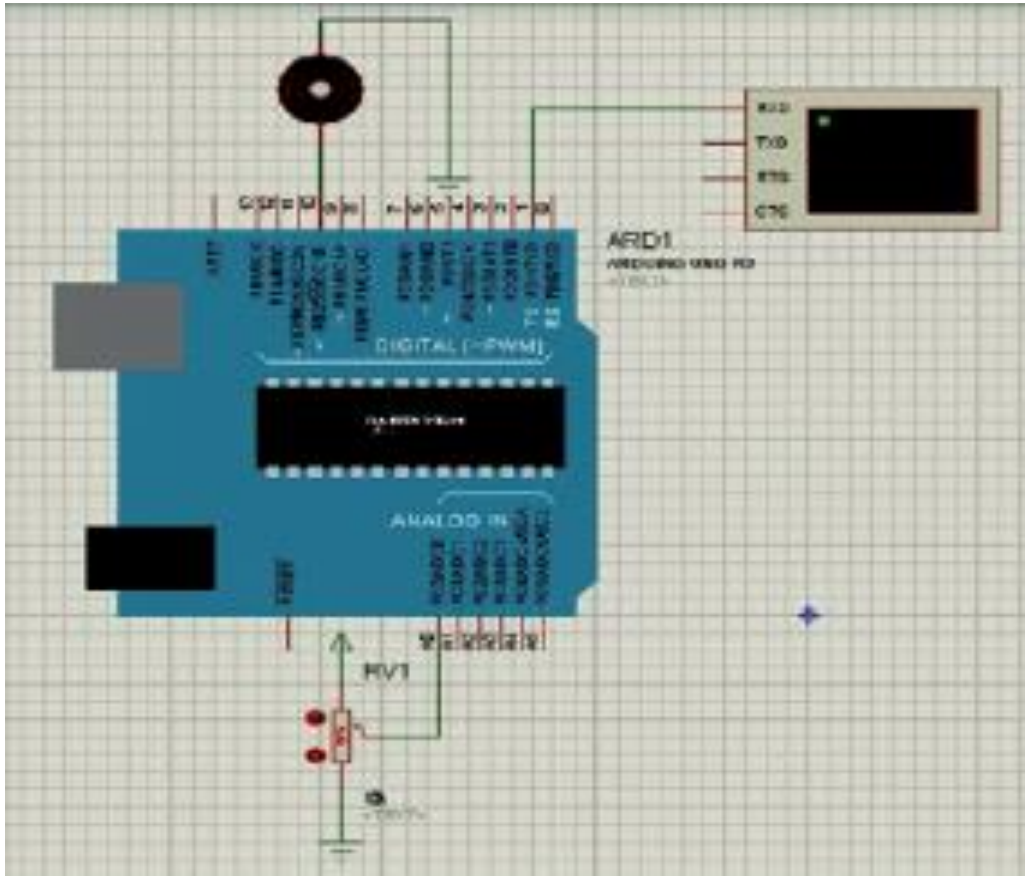
PWM (Control De Ancho De Pulso)

Programación En Arduino

```
        /*  CONVERSOR ANALOGO DIGITAL  */
int valor_analogo; //Inicialización de variable
void setup() {
  pinMode(led10, OUTPUT); //Pin declarado como salida
  Serial.begin(9600); //Configuración de cx serial
}

void loop() {
  valor_analogo=analogRead(); //Lectura de canal análogo
  analogWrite(10,valor_analogo/4); //Enviar a pin digital
  Serial.println(valor_analogo/4); //Impirmir mensaje
}
```

PWM (Control De Ancho De Pulso)

Simulación



EJERCICIOS

Cambios de velocidad de motor

Descripción

El programa cambiara la velocidad del motor dependiendo cuantas veces se presiona la letra A, y disminuirá su velocidad cuando se presione la letra B, con un máximo de 5 veces cada una.

1.9 EJERCICIOS

Cambios de velocidad de motor

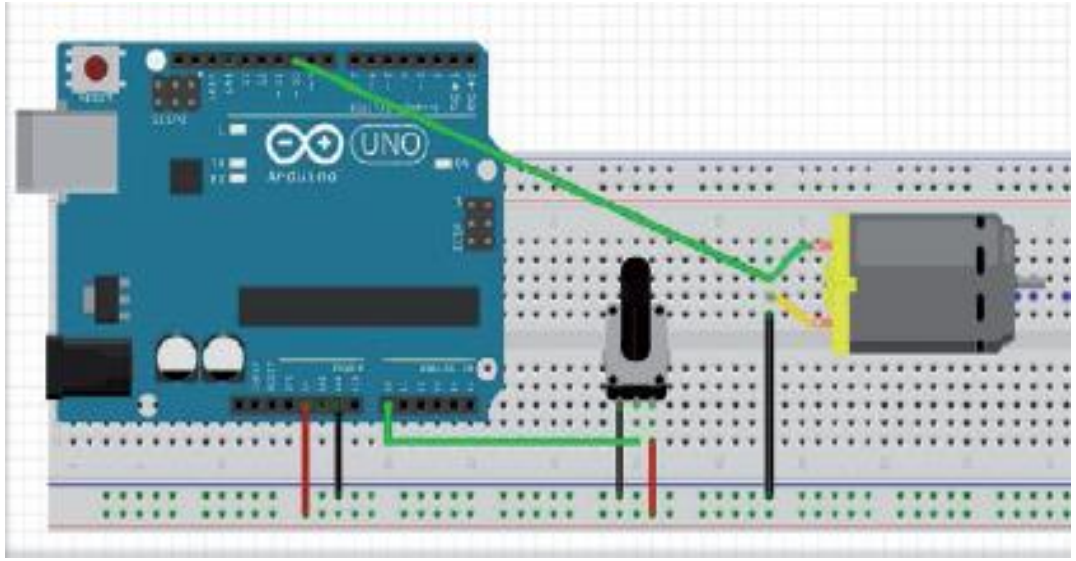
Programa En Arduino

```
char val
int valor_pwm = 0;
int motor = 7;

void setup() {
  pinMode(22, OUTPUT);
  pinMode(motor, OUTPUT);
  Serial1.begin(9600);
}

void loop() {
  if (Serial1.available() > 0) { //Puerto recibe dato
    val = Serial1.read(); //Lee el dato de la cx serial
    if (val == 'A') { //Condicion de dato de entrada
      if (valor_pwm < 250) { //Condicion de velocidad
        valor_pwm = valor_pwm + 50; //Incrementar la velocidad cada vez en 10%
      }
    }
    if (val == 'B') { //Condicion de dato de entrada
      if (valor_pwm >= 50) { //Condicion de velocidad
        valor_pwm = valor_pwm - 50; //Decremento de velocidad en 10%
        analogWrite(motor, valor_pwm); //Enviar dicho valor al pin
      }
    }
  }
```

EJERCICIOS

Cambios de velocidad de motor

Simulación



EJERCICIOS

1.9.2 Control de Giro de Servo Motor

Descripción

El servo motor su funcionalidad es que no gira 360 grados libremente como un motor normal, es un motor con engranes para que tengan mayor fuerza y se pueda girar dependiendo de las necesidades del usuario, de manera general se usa el HS-311.

EJERCICIOS

Control de Giro de Servo Motor

Programa En Arduino

```
#include <Servo.h> //Libreria
Servo myservo; //Crear objeto

int potpin = 0;// Canal analogo para usar
int val; //Variable de lectura del analogo

void setup() {
  myservo.attach(9); //Pin de conexion del servo
}

void loop() {
  val = analogRead(potpin); //Lectura de valor analogo de 0 a 1023
  val = map(val, 0, 1023, 0, 180) //Escala de uso del servo de 0 a 180 cambiado escala del analogo digital
 myservo.write(val); //Enviar valor del analogo al servo
  delay(15); //Tiempo de espera del servo
}
```

EJERCICIOS

Control de Giro de Servo Motor

Simulación