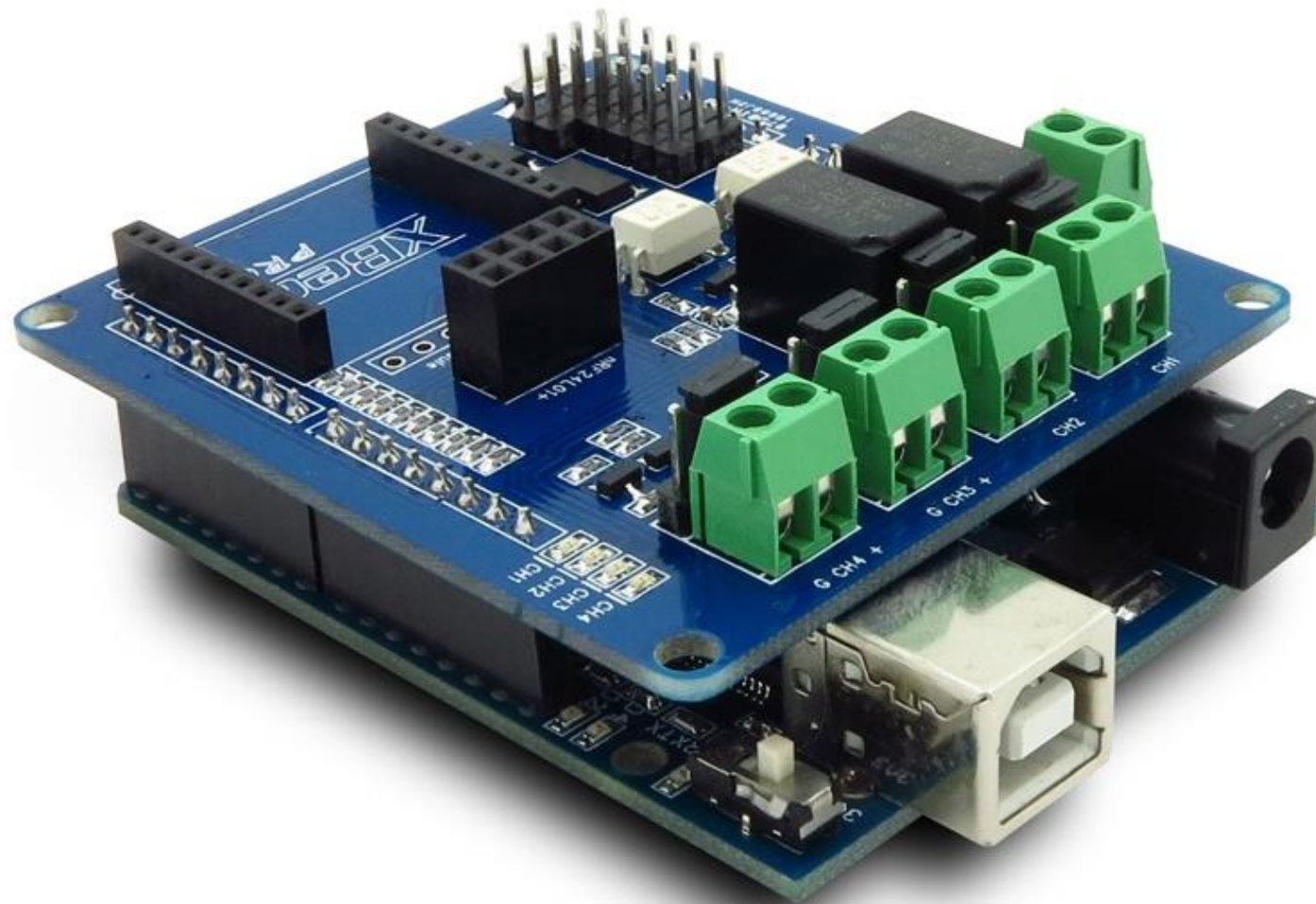




CAD-CDA

SISTEMAS EMBEBIDOS



Entradas Analógicas

Las tarjetas de arduino, generalmente cuentan con 6 entradas analógicas (A0 – A5).

Estas entradas funcionan como convertidores Analógico a Digital (ADCs) **de 10 bits.**

El voltaje de referencia de las entradas analógicas se puede ajustar mediante el pin Vin y el jumper Vin.

El jumper Vin es utilizado para configurar un voltaje de referencia de 5V



Cuidado: Si el jumper Vin está conectado y en el pin Vin se conecta otra fuente de poder, podría dañar la tarjeta.

analogRead()

analogRead()

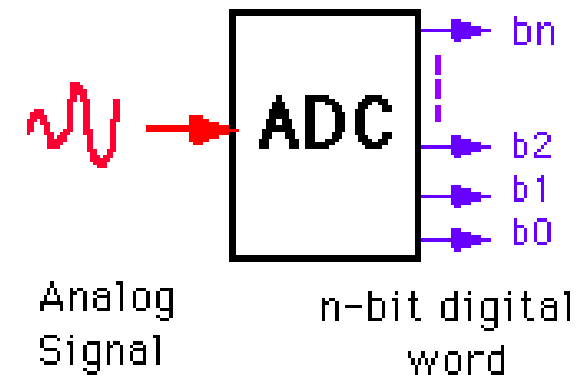
Esta función es utilizada para leer un valor analógico de alguna de las terminales de entrada A0 a A5. Este valor es representado por 10 bits, arrojando valores enteros (**int**) entre 0 y 1023.

Esta función tarda cerca de 100 microsegundos para leer una entrada analógica, por lo que el máximo número de lecturas es de 10,000 por segundo.

Sintaxis:

```
analogRead(pin)
```

Donde **pin** es el número de la terminal analógica a leer (A0 – A5)



Código

AnalogInput.ino

```
int sensorPin = A0;  
int ledPin = 13;  
int sensorValue = 0;
```

Se declara una variable que representará la terminal analógica A0.

Variable que recibirá el valor analógico representado por 10 bits.

```
void setup() {  
  pinMode(ledPin, OUTPUT);  
}
```

La terminal 13 se configura como salida.

```
void loop() {  
  sensorValue = analogRead(sensorPin);  
  digitalWrite(ledPin, HIGH);  
  delay(sensorValue);  
  digitalWrite(ledPin, LOW);  
  delay(sensorValue);  
}
```

Lectura de la terminal analógica A0.

Se cambia el estado del pin de salida entre ALTO y BAJO.

Retardos que dependen del valor leído de la terminal analógica





Salidas Análogas

Se puede generar salidas análogas tan pronto como sale de su caja, y no requiere hardware extra.

Las salidas análogas son generadas por medio de señales **PWM**.



No todas las terminales de Arduino son capaces de generar una salida análoga, solo aquellos que están marcados con una tilde (~) junta a su identificador de terminal, son capaces de generar este tipo de señal.

analogWrite()

La llamada a esta rutina indica a una terminal de salida que debe de generar una señal PWM de manera continua con el ciclo de trabajo indicado

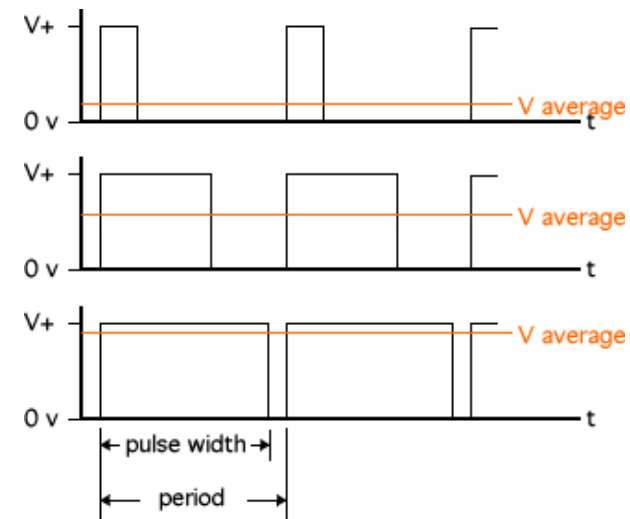
La señal PWM genera con esta rutina se mantiene de **manera continua** en la terminal de salida hasta que se escribe un nuevo analogWrite() a la misma terminal, o se ejecuta un digitalWrite() o digitalRead().

Sintaxis:

analogWrite(pin, CICLO)

CICLO: 0 a 255.

analogWrite()



PWM puede utilizarse para el control de servo motores, motores de corriente directa o para regular la intensidad con la que brilla un led

Código fade.ino

```
int led = 9;  
int brightness = 0;  
int fadeAmount = 5;
```

```
void setup() {  
  pinMode(led, OUTPUT);  
}
```

Terminal 9 toma el rol de salida.



```
void loop() {  
  analogWrite(led, brightness);  
  brightness = brightness + fadeAmount;
```

Escribir la cantidad de "brillo" que queremos obtener del led. En este caso el 100% de ciclo de trabajo esta representado por 255.



```
    if (brightness == 0 || brightness == 255)  
        fadeAmount = -fadeAmount;
```

```
    delay(30);  
}
```

Una vez que se alcanza alguno de los valores máximos, se invierte de signo para alcanzar el valor opuesto de "brillo"



control Servos



Que es un Servo Motor?

Un servo, es un pequeño motor que puede colocar su eje en una posición deseada y mantener esa posición.

Las instrucciones al servomotor son enviadas en forma de señales PWM. Donde diferentes ciclos de trabajo indican diferentes posiciones la Servo motor.

Biblioteca Servo.h

La biblioteca Servo.h permite manejar de manera simple un servo motor.

Sin embargo requiere una actualización para mejorar la experiencia en el desplazamiento del motor.

Servo myservo()

La librería Servo es una librería orientada a objetos, cada servo a controlar debe estar ligado a un objeto del tipo Servo.

Cada objeto es independiente.

Sintaxis:

Servo (ID)

ID: Identificador del Servo.

myservo.attach()

La rutina attach() debe ser invocada con un objeto del tipo servo. Esta rutina indica en que pin de Galileo se encuentra el servo motor físicamente.

Sintaxis:

myservo.attach(PIN)

PIN: Terminal a la que se encuentra conectado el servomotor.

myservo.write()

Por medio de esta función se indica la posición que el servo motor debe de tomar.

Sintaxis:

myservo.write(VALUE)

VALUE: Posición del servo motor.

Código servo.ino

`#include <Servo.h>` ← Importar la librería Servo.h

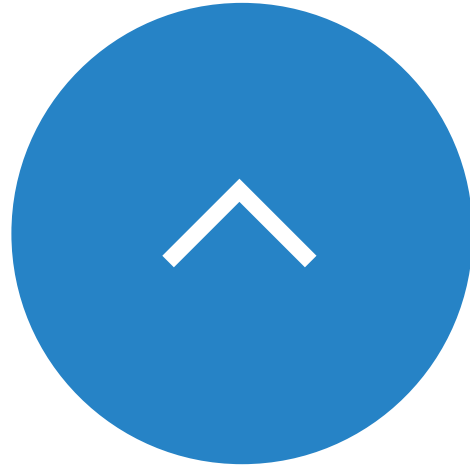
`int servoPin = 9;`

`Servo myServo;` ← Creación de un objeto de la clase Servo. Igual que en programación orientada a objetos

`void setup() {`
 `myServo.attach(servoPin);` ← Asignar el pin 9 al objeto Servo. Todas las operaciones sobre el objeto 'myservo' se reflejarán en el pin 9.
`}`

`void loop() {`
 `myServo.write(19);` ← Envío de señal de control al servo motor para asignar una nueva posición.
 `delay(1000);`
 `myServo.write(180);`
 `delay(1000);`
`}`

PROGAMACIÓN



CÓDIGO 14: CAD Y PWM



CÓDIGO 15: CAD Y SERVO