Programación para Sistemas Embebidos

4. Estructuras de control y Funciones

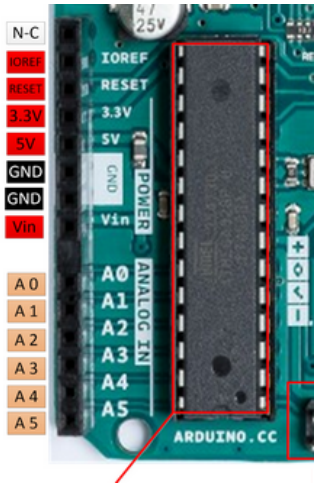
4.1. Practico 04 - Pines analógicos

Ya sabemos que Arduino puede manejar valores digitales como entrada y como salida y tomar decisiones con esta información. Pero en nuestro entorno, existen de manera frecuente variables continuas, por ejemplo temperatura, humedad, etc. que toman valores en todo un rango de posibilidades.

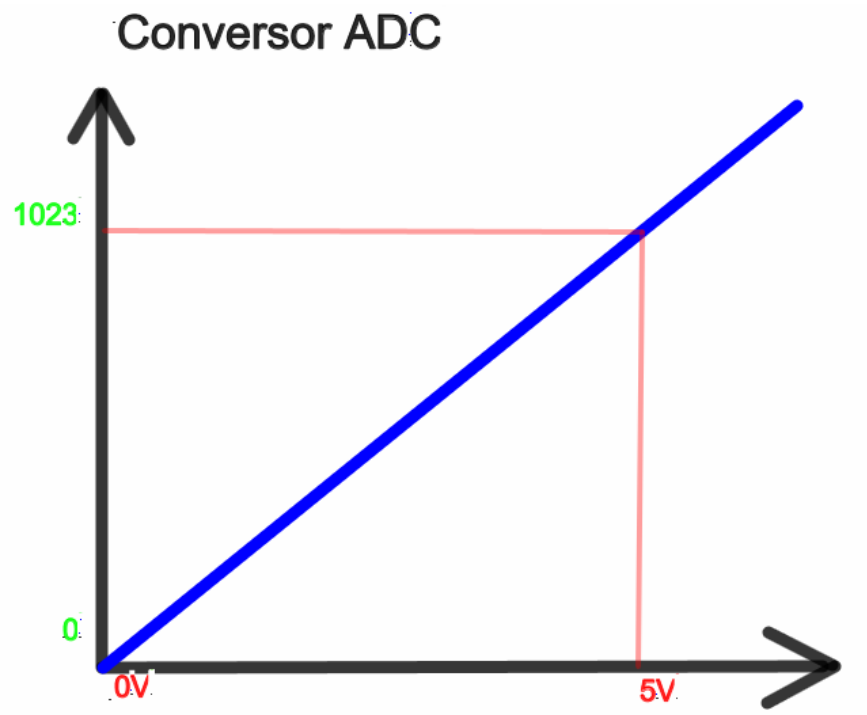
Nuestra placa Arduino, también está preparada para manejar esta información y en este práctico veremos como hacerlo.

Entradas analógicas

Cuando revisamos el PINOUT de la placa, nombramos estos pines que se denominan entradas analógicas o en inglés Analog Inputs



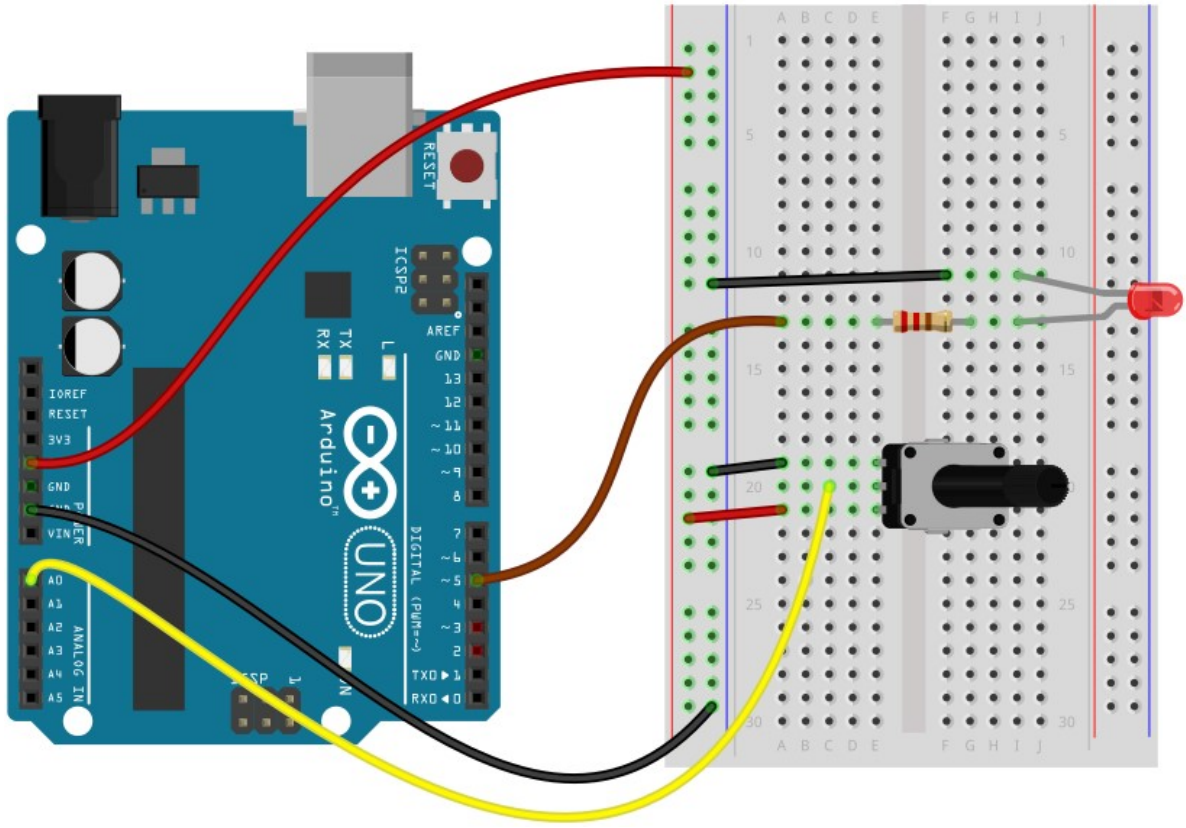
En ellas podemos conectar un sensor con salida entre 0 y 5 V, pero a diferencia de las entradas digitales, la señal de entrada puede tener cualquier valor, por ejemplo 3,1 V.



De este modo, conectando una tensión de 0V en un pin analógico, el ADC lo convertirá al número entero 0. Si colocamos una tensión de 5V, la conversión dará como resultado el número 1023.

Y para todo el rango intermedio, la tensión que conectamos aquí va a ser transformada a un número entre 0 y 1023.

El circuito a construir sería como el siguiente:



Con una resistencia ajustable o potenciómetro, podemos ajustar la tensión que conectamos en A0 entre Vcc y GND (5V - 0V).

Variables INT

Nuestro programa debe leer ese valor y almacenarlo en una variable, por esto, vamos a crear una variable tipo INT que puede guardar números enteros.

```
int tension = 0;
```

y luego leer el valor con la instrucción analogRead:

```
void loop(){
    tension = analogRead(A0);
}
```

Si bien este programa es funcional, no podemos saber lo que está haciendo.

Nuevamente haremos uso de nuestro LED conectado en la salida para ver como varía este valor.

Agreguemos las siguientes lineas para hacer parpadear el LED con una cadenacia proporcional a la tensión en la entrada A0.

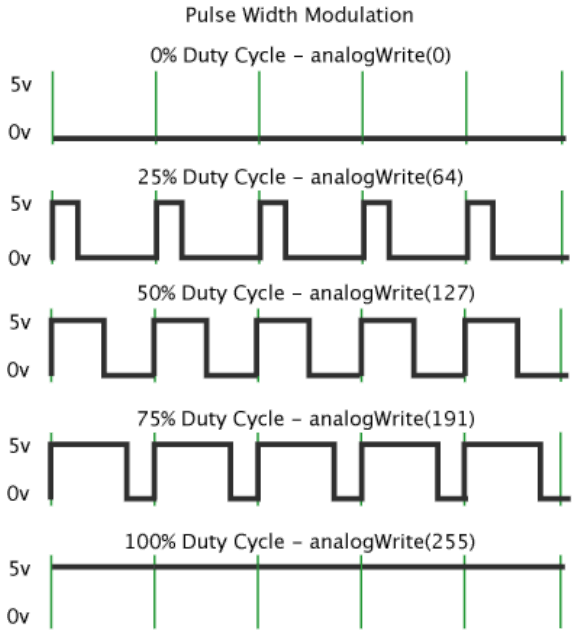
```
int tension = 0;
void setup() {
  // put your setup code here, to run once:
  pinMode(5, OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  tension = analogRead(A0);
  analogWrite(5, tension);
}
```

Probemos variar el potenciómetro y veamos que sucede cuando ponemos tensiones bajas en la entrada.

Si el valor de la resistencia es bajo, vemos un brillo intermedio, esto sucede porque el LED prende y apaga más rápido de lo que nuestros ojos pueden detectar los cambios, por lo que lo vemos con un brillo intermedio. Si es un poco mayor, podemos ver que varía la velocidad de parpadeo.

Modulación por Ancho de Pulso (PWM)

Usando esta idea podemos conseguir valores "intermedios" en las salidas digitales, prendiendo y apagando la salida un cierto porcentaje del tiempo, y haciéndolo rápido, el brillo del led parecerá que varía de manera analógica. Esta técnica se llama Modulación por ancho de pulso.



Arduino tiene algunos pines digitales con la posibilidad de configurarlos como salidas PWM, en la placa está impresa la tilde.

Para esto debemos usar la instrucción analogWrite, el pin en el que se va a escribir y el valor.

Pero primero, tenemos un problema.

Ya dijimos que la conversión del ADC nos entregará un valor entre 0 y 1023, pero los pines PWM trabajan con valores entre 0 y 254, por lo que deberíamos escalar el rango entre una y otra escala.

Por esto al escribir en la salida, dividiremos la variable tensión por 4.

```
void loop() {
  // put your main code here, to run repeatedly:
  tension = analogRead(A0);
  analogWrite(5, tension/4);
}
```

Función map

El entorno de Arduino, tiene una función llamada map, que sirve para escalar entre dos rangos de valores, funciona del siguiente modo:

Syntax

```
map(value, fromLow, fromHigh, toLow, toHigh)
```

Podríamos conseguir el mismo funcionamiento realizado anteriormente, escribiendo:

```
void loop() {
  // put your main code here, to run repeatedly:
  tension = analogRead(A0);
  analogWrite(5, map(tension, 0, 1023, 0, 255));
}
```

Ejercicios

1. Escribir un programa que haga parpadear al LED en intervalos de 1 a 5 segundos según la posición en la que esté el potenciómetro.

2. Escribir un programa que además de variar el brillo del LED, lo haga parpadear en intervalos de 1 segundo a 5 segundos.

Fuentes:

https://www.arduino.cc/en/Tutorial/BuiltInExamples/AnalogInput

https://www.arduino.cc/en/Tutorial/BuiltInExamples/Fading

https://docs.arduino.cc/learn/microcontrollers/analog-output

https://www.arduino.cc/reference/en/language/functions/math/map/