ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Mecánica y Ciencias de la Producción

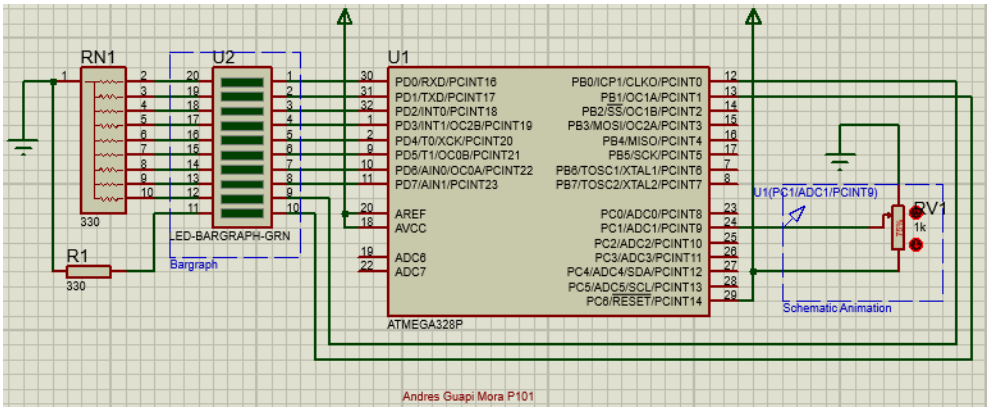
PRACTICA #6

Fecha: 21/06/2021

Oscar Andres Guapi Mora
oguapi@espol.edu.ec

ADC

Diagrama



Código

main.c

```c
#include "config.h"
#include "funciones.h"

int main()
{
    conf();
  while (1)
      {
    lectADC();
    PORTD = ADCL;//Visualizaremos en el bargraph
    PORTB = ADCH;
    _delay_ms(200);
  }
    return 0;
 }
 //Andres Guapi P101
```

config.c

```c
#include "config.h"

void conf(){
      //LLamar configuracion
    DDRD = 0XFF;              //todo sera salida
    DDRB |= ((1<<0)|(1<<1));//solo el bit 1 y 0 seran salida
    ADMUX = 0B01000001;   //AVCC- AREF y ADC1 configuramos al la entrada 1
    ADCSRA = 0B00000111 ;//PRE-SCALER  128 - DISABLED ANALOG CONVERTER & CONVERSION
                         //NOT STARTED, 8MHz/128
    ADCSRB = 0B00000000; //FREE RUNNING MODE
    DIDR0 = 0B00000010;    //ENABLE ANALOG MODE ADC1 desabilitando el modo digital

}
//Andres Guapi Mora
```

funciones.c

```c
#include "funciones.h"
//Andres Guapi

void lectADC(){
   //Durante la ejecucion, llamar las funciones
   ADCSRA = 0B11000111 ; // ENABLED ANALOG CONVERTER & START CONVERSION, bit 7 activacion
                         //y 6 dar inicio
   while(ADCSRA & (1<< ADSC)); // ADSC DISABLED? Si es 1 sigue haciendo la conversion
   //ADCSRA = 0B00000111;//puedo inhabilitarlo pero no es necesario
}
```

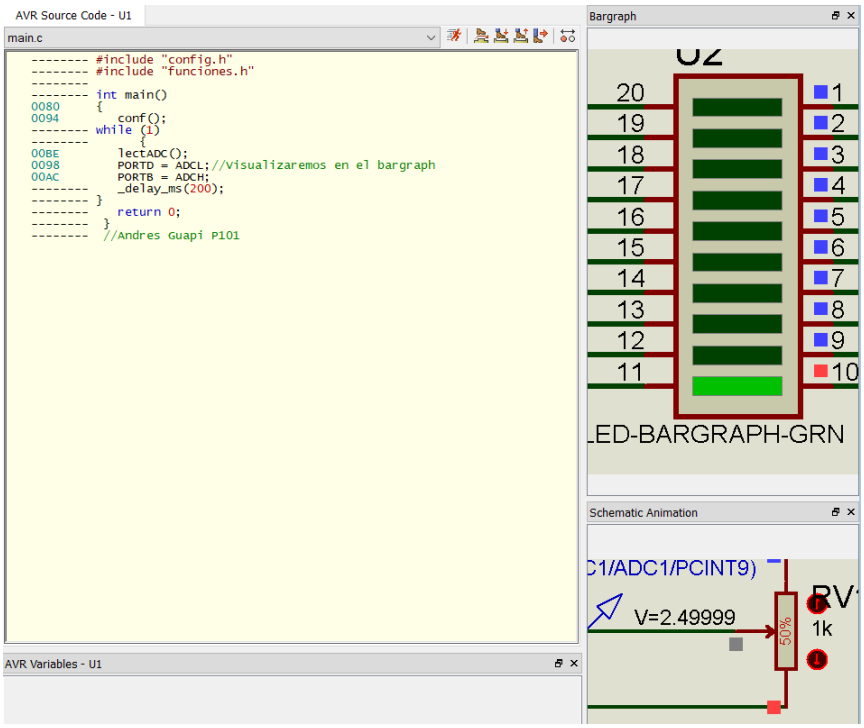


Ilustración 1: Simulación con el potenciómetro al 50%.

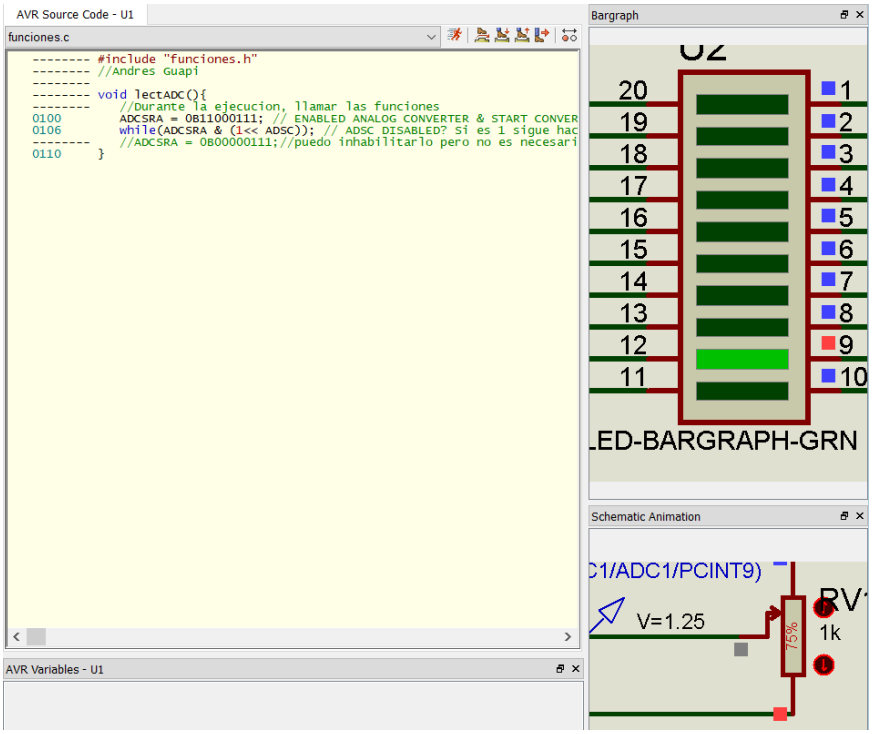


Ilustración 2: Simulación con el potenciómetro al 75%.

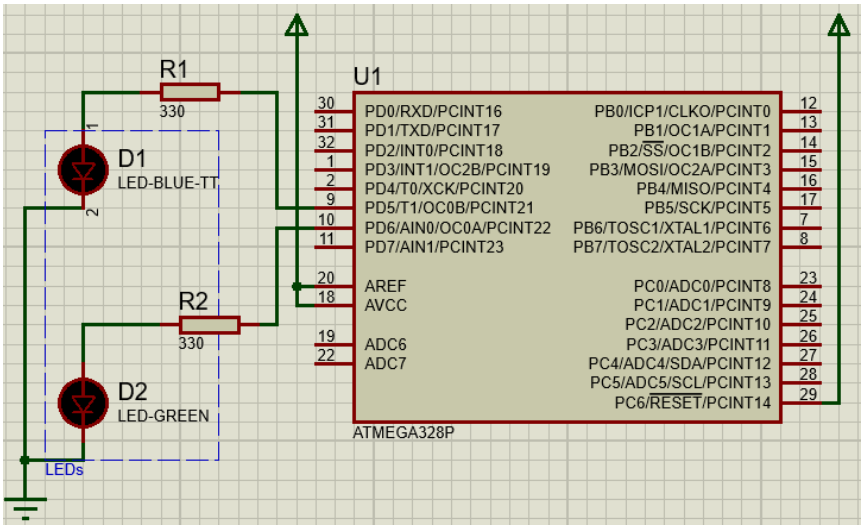
Para conocer el numero binario que nos da el ADC primero debemos aplicar la siguiente formula:

$$INT(N) = \frac{V_{in}x(2^n - 1)}{V_{ref}}$$

Aplicando la formula anterior para el caso de la Ilustración 1: Simulación con el potenciómetro al 50%. (50%) tenemos un resultado de 512, el cual pasado a binario nos da 1000000000 que es lo mostrado en el Bargraph. En el caso de la Ilustración 2 para 75% en cambio al aplicar la formula anterior tenemos un resultado de 256 el cual pasado a binario es 100000000 que es lo mostrado en el Bargraph.

PWM

Diagrama



Codigo

main.c | config.c | funciones.c

```c
#include <avr/io.h>
#include <util/delay.h>

int main(void){
    conf();
while(1){
      incrementoTimer();
      _delay_ms (2000L);
      disminucionTimer();
      _delay_ms (3000L);
    }
   return 0;
}
//Andres Guapi P101
```

ain.c | *config.c | funciones.c | config.h | funciones.h

```
1  #include "config.h"
2
3  void conf(){
4     DDRD |= 0b01100000; //SALIDA PINES PD5(OC0B) Y PD6(OC0A),bits de salida PWM
5     //Usamos el Timer 0
6     TCCR0A = 0b11100011 ; //TIMER0 PWMA// INVERTIDO A & NO INVERTIDO B &MODO RAPIDO,
7                           //cofiguro los pines 7 y 6 1,1 (invertido); pines 1,0(modo rapido)
8     TCCR0B = 0b00000001; //SIN PREESCALADOR
9     OCR0A = 0; //valor inicial de pwm para el pin OC0A, como esta invertida esta senal iniciara en 1 o 254, color verde
10    OCR0B = 0; //valor inicial de pwm para el pin OC0B color azul
11 }//Pinb es azul
12
13 //Andres Guapi Mora P103
```

in.c | config.c | funciones.c | co

```
1  #include "funciones.h"
2  //Andres Guapi Mora
3  void incrementoTimer(){
4     do{
5          //Regularemos la intensidad de la senal
6          OCR0A++;       //Del Timer 0 el PWM A
7          OCR0B++;       //Del timer0 el PWM B
8          _delay_ms (10);
9     }while(OCR0A<255);
10 }
11
12 void disminucionTimer(){
13    do{
14    //disminuye 1 y 1
15    OCR0A--;
16     OCR0B--;
17         _delay_ms (10);
18    }while(OCR0A>0);
19 }
```

Simulación

main.c

```
-------- #include <avr/io.h>
-------- #include <util/delay.h>
--------
0080     int main(void){
008C        conf();
-------- while(1){
0090           incrementoTimer();
--------       _delay_ms (2000L);
00A0           disminucionTimer();
--------       _delay_ms (3000L);
--------    }
--------  return 0;
-------- }
-------- //Andres Guapi P101
```

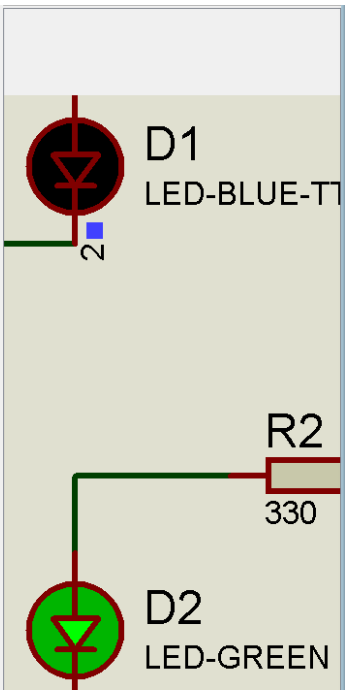


Ilustración 3: Incremento del timer con PWM invertido (D1) y PWM no invertido (D2).

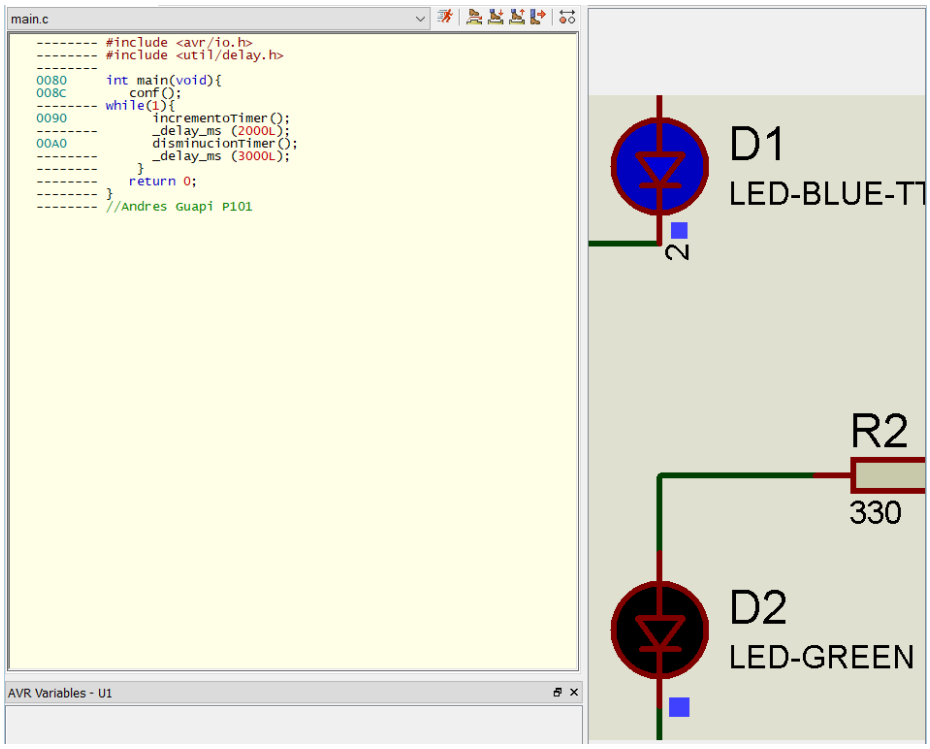


Ilustración 4:Decremento del timer con PWM invertido (D1) y PWM no invertido (D2).

A ambos leds se le dio el valor de 0, pero el led verde inicia encendido como vemos en la Ilustración 3 ya que es el PWMA del timer0 (OCR0A) fue configurado invertido, en cambio el led azul al ser configurado como no invertido se iba encendiendo. Por otro lado, cuando disminuimos los timer Ilustración 4 pasa lo contrario el led azul se iba encendiendo y el verde se iba apagando.

Conclusiones

- Fue posible regular la intensidad de dos leds de diferentes formas al configurar las salidas PWM del microcontrolador Atmega328p, donde vimos que a pesar de las dos salidas iniciar en cero como uno estaba invertido (led verde) iniciaba encendido y el otro iniciaba apagado por estar en modo no invertido.
- Pudimos configurar el ADC del microcontrolador Atmega328p mediante la manipulación de registros ADMUX el cual establecía el voltaje de referencia con el que trabajaría el ADC, además en el mismo registro con los 4 bits menos significativos podíamos elegir con cual entrada analógica íbamos a trabajar.
- Mediante un bargraph pudimos visualizar la señal digital proveniente del registro ADCH y ADCL del ADC que adquirió una señal analógica proveniente de un potenciómetro, además de como ir del voltaje de entrada al valor ADC y viceversa.

Recomendaciones

- Si presenta errores en el Proteus como cambio de nombre en los archivos unos con otros, la mejor opción sería cerrar el programa y volverlo abrir porque seguro son errores del propio programa.
- Es recomendable que en el archivo con las funciones asegurarse de llamar el archivo donde se encuentran creadas las variables del proyecto para evitar errores por no declaración de variables.