LED RGB Y RES LDR

Utilización de PWM y ADC

Universidad Nacional de La Plata
Facultad de Ingeniería

Circuitos Digitales y Microcontroladores
TP n°4

Pérez, Francisco
Gallo, Molinuevo Francisco

Problema

Realizar un programa para controlar la intensidad y el color del LED RGB con la técnica de PWM. En el kit de clases el mismo se encuentra conectado a los terminales PB5, PB2 y PB1 (RGB) a través de resistencias de limitación de 220ohms y en forma ánodo común.

Introducción

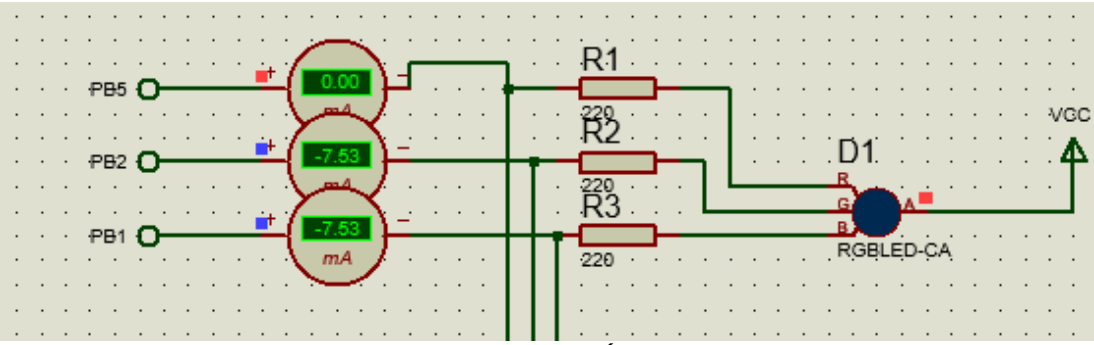
En este entregable se tiene como objetivo utilizar funciones de los temporizadores que, en trabajo anteriores no se utilizó. Por otro lado, un nuevo periférico va a ser implementado, siendo este el ultimo que faltaba manejar siendo el conversor analógico digital. Para poder utilizar este periférico, se debe tener en posesión un componente que genere un voltaje (analógico) para que este lo pueda convertir a un valor digital. Este componente es la resistencia LDR.

1. Componentes:

La placa que provee la catedra contiene varios componentes donde se puede utilizar, entre ellos está el led RGB y la resistencia LDR

1.1 Led RGB:

El led RGB es un componente electrónico que emite luz en los tres colores primarios (Rojo, verde y azul) donde, estos tres colores combinados en diversas proporciones crean una gama de colores visibles, esta proporción es explicada más adelante.
El led está compuesto por 3 diodos, uno para cada color, el cual se activan individualmente, la forma en que está conectado es en ánodo común *(ver conexionado 1)*



Conexionado 1 RGB-Ánodo común

Cada uno de estos diodos tiene una resistencia de 220Ω

El objetivo de este proyecto es de hacer parpadear un color en particular en un periodo dado.

1.2 Resistencia LDR:

La resistencia LDR es un fotorresistor siendo un tipo especial de resistencia el cual el valor de esta cambia en función a la cantidad de luz que incide sobre esta. La resistencia es alta en condiciones de oscuridad y baja en exposición a la luz, es decir, el componente genera un voltaje mayor cuando está en oscuridad y un voltaje menor cuando está expuesto a la luz.
Como el voltaje es una señal analógica, es esperado tener que convertir tal señal en una digital y cuantificar este valor que nos da la resistencia para poder realizar decisiones en base a este valor. Se desarrollará más adelante en el informe.

2. Temporizadores:

Se hará uso de dos temporizadores que contiene el MCU Atmega328p, siendo estos el TIMER 0 y TIMER 1. Uno de ellos se encargará de temporizar toda la solución y otra, la generación de señales PWM para el control del componente Led RGB.

2.1 Timer 0:

Este periférico de 8 bits es utilizado para poder generar una interrupción periódica, esta interrupción levantara varios flag que son utilizados para que luego una función visualice los flag y mande a realizar acciones. Una de las acciones es actualizar la máquina de estados finitas encargada de llevar el funcionamiento correcto del parpadeo del LED con el periodo que indica el enunciado, el otro flag controlara el PWM generado por software.

2.1.1 Configuración:

Este Timer será configurado tal de que genere una interrupción cada 8 us, para esto se deben modificar los registros de configuración. El temporizador funcionara en modo CTC con generación de interrupción cuando el registro de comparación con el contador sea el mismo.

Los registros a modificar son:

TCCR0A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
0x24 (0x44)	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	TCCR0A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

TCCR0B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
0x25 (0x45)	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	TCCR0B
Read/Write	W	W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Donde de WGM02, WGM01 y WGM00 determinan el modo de operación, como se mencionó, será en modo CTC. Para esto, los valores deben ser WGM01 = 1 y los demás en 0. En c, esto se traduce a `TCCR0A |= (1<<WGM01);`

Como la interrupción es generada cada vez que el valor del contador llega al valor de comparación, se debe buscar los valores exactos de estos.

La ecuación de la frecuencia generada por el temporizador es:

$$f_{OCnx} = \frac{f_{clk\_I/O}}{2xNx(1 + OCRnx)}$$

Siendo N el preescalador aplicado y OCRnx el valor del registro de comparación.

Sabiendo que $f_{clk\_I/0} = 16MHz$ y $f_{OCnx} = \frac{1}{8us} = 125000\ Hz$

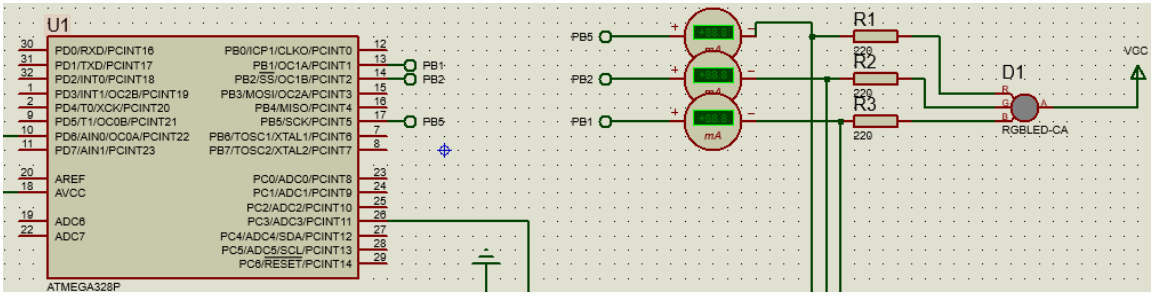
Siendo N=1, el valor del OCRnx para que la rutina de interrupción se ejecute cada 8 us:

$$(OCRnx) = \frac{f_{clk\_I/O}}{2.N.f_{OCnx}} - 1 = 63$$

Para aplicar un preescalador, se deben modificar los registros CS02, CS01, CS00. En este caso se optó por aplicar uno de 1, por lo que estos toman los valores de CS00=1. En c, `TCCR0B |= (1<<CS00)`

2.2 Timer 1:

Uno de los aspectos a tener en cuenta al realizar la solución es el conexionado físico de la placa que provee la catedra, el cual el led RGB está conectado a los terminales PB5, PB2 y PB1. Donde PB2 y PB1 son los canales de salida del TIMER1. El PB1 le corresponde a la señal generada por el registro OC1A y el PB2 al registro OC1B (ver conexionado 2). Es decir, se parte sabiendo que los colores verde y azul se controlan mediante el Timer 1.



Conexionado 2 conexión de pines RGB al MCU

2.2.1 Implementación:

Se utilizará el modo Fast PWM 8-bit para los 2 registros A y B del Timer 1.

2.2.1.1 PWM:

PWM significa Modulación por Ancho de Pulso, esta técnica controla la potencia de una señal digital. Esta permite ajustar la cantidad promedio de energía entregada a un componente mediante la variación del ancho de los pulsos en una señal cuadrada.

La señal cuadrada es generada con un ciclo de trabajo (duty cycle) que representa la proporción de tiempo que la señal esta encendido o en alto en relación al ciclo total. Es decir, si por ejemplo una señal cuadrada tiene un ciclo de trabajo del 30%, significa que la señal está en alto durante 1/3 del tiempo. (ver representación 1)

Representación 1

En modo Fast PWM 8-bit existen 2 modos de operación. El invertido y no invertido. Estos modos determinan cómo se comporta la señal de salida.

El modo no invertido la señal permanece en bajo el porcentaje del ciclo de trabajo especificado y en alto el resto. Es decir, si el ciclo de trabajo es del 30%, durante este porcentaje estará apagado y durante 70% encendido.

Mientras tanto, el invertido es lo contrario.

Estos 2 modos tienen 2 limitaciones cada uno, en el modo invertido, no se puede alcanzar el valor mínimo de 0, mientras que el no invertido si. Por otro lado, el no

invertido no puede alcanzar el valor máximo (255) pero si el invertido (ver ecuaciones a y b). Para este entregable, se optó utilizar el modo no invertido por el hecho de poder llegar a ese valor mínimo de 0 (ciclo de trabajo 0%) por lo que el led se apagara completamente, en el modo invertido esto no sería posible

Al tener un registro de 8 bits, se pueden generar 255 ciclos de trabajos distintos y estará dado por el valor en el OCRNX, un valor más cercano a 255 significará un ciclo de trabajo bajo. Mientras que, si nos acercamos a 0, el ciclo de trabajo es mayor.

$$duty\ cycle = \frac{OCR1+1}{256} X100$$ ***ecuación a*** *ciclo de trabajo modo no invertido*

$$duty\ cycle = \frac{255-0CR1}{256} X100$$ ***ecuación B*** *ciclo de trabajo modo no invertido*

2.2.1.2 Configuración:

Se debe modificar los registros de configuración de modo que este en modo Fast PWM 8-bit y que genere la señal por los terminales de salida.

Para el modo Fast PWM 8-bit debemos modificar los bits WGM10 y WGM12, estando en 1 los dos, en C: `TCCR1A |= (1<<WGM10) TCCR1B |= (1<<WGM12)`

Luego, se aplicará un preescalador de N=1024 obteniendo una frecuencia de 61 Hz (ver ecuación 1), modificando los bits CS01 y CS00 `TCCR1B |= (1<<CS01) | (1<<CS00)`

$$f_{generated\_wave} = \frac{f_{oscillator}}{256.N}$$ *Ecuación 1*

Para finalizar la configuración, debemos especificar como generar esta señal en el canal de salida, en modo Fast PWM hay varias maneras (ver figura 2)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on compare match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on compare match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on compare match, clear OC1A/OC1B at BOTTOM (inverting mode)

Figura 2 generación de señal

Como utilizamos modo invertido, seteamos estos 2 bits en uno. Recordar que se va a utilizar los dos pines de salida que contiene ya que necesitamos señales PWM independientes para cada color por lo que se modifican COM1A1, COM1A0, COM1B1 y COM1B0.

```
TCCR1A |= (1<<COM1B1)
TCCR1A |= (1<<COM1A1)
```

Ahora el Timer 1 está listo para generar señales PWM independientes para el verde y el azul. La forma de generar un color deseado es modificando los registros OC1A y OC1B, que, como se explicó anteriormente, un valor alto en estos significa menos intensidad de ese color y un valor bajo, mayor intensidad.

El pin PB2 (OC1B) le corresponde al verde y el pin PB1(OC1A) al azul. (Ver imagen 1)

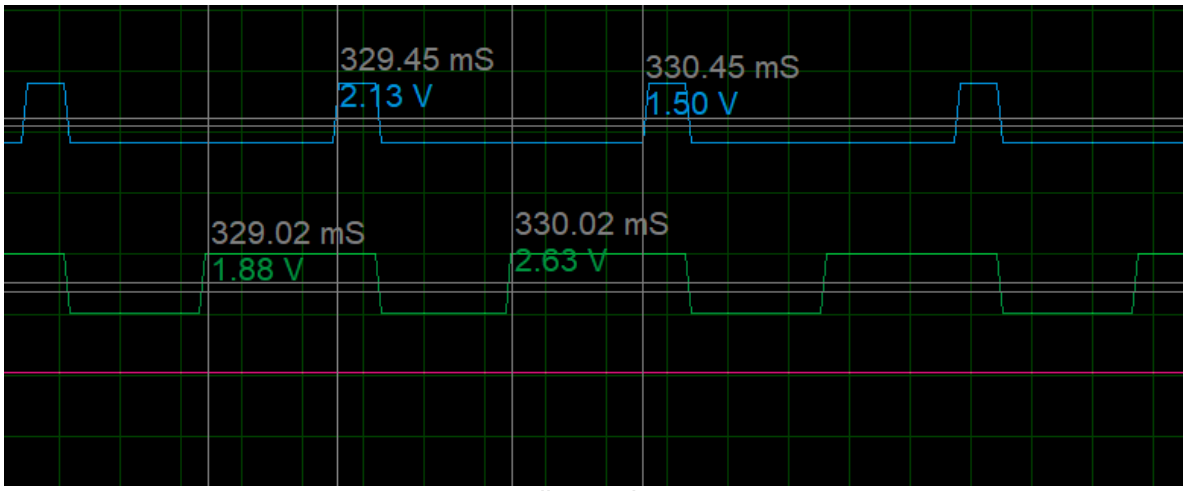


Imagen 1 osciloscopio PWM

2.2.1.3 PWM por software:

Al tener los pines de led ya colocados físicamente, existe esta limitación de no poder utilizar otro pin de un timer para poder generar otra señal PWM, en este caso, para el led rojo. Como esto no es posible, existe una manera de generar PWM por software y colocarla en la salida que le corresponde al rojo y suplir este problema.

Para esto, se utilizará la interrupción que genera el timer 0 como frecuencia de actualización del PWM.

Por otro lado, se definirán varios valores para la legibilidad del programa

```
#define PWM_PERIOD 255
static uint8_t PWM_DELTA = 0;
#define PWM_OFF PORTB |= (1<<PORTB5)
#define PWM_ON PORTB &=~ (1<<PORTB5)
```

Donde PWM_OFF y ON escribe un 1 o 0 como salida, como es anodo común, un 0 es prendido y un 1 apagado.

PWM_PERIOD es una constante que define el total del periodo y donde modificando PWM_DELTA estaremos generando un ciclo de trabajo. Es decir, si PWM_DELTA es 255 (PWM_DELTA = PWM_PERIOD), el duty cycle es 100%.

La rutina de interrupción del Timer 0 levantara un flag indicando la actualización de la función de PWM por software, siendo esta:

```
void PWM_soft_update(){
	static uint16_t PWM_position=0;
	if(++PWM_position >= PWM_PERIOD){
		PWM_position = 0;
		PWM_OFF;
	} else {
	if (PWM_position < PWM_DELTA){
		PWM_ON;
		} else {
		PWM_OFF;
	}
```

```
        }
    }
```

En donde se puede visualizar que la salida estará en alto hasta que llegue al valor de PWM_DELTA, siendo el ciclo de trabajo. El resto del periodo estará en bajo

3. Analog to Digital Converter:

3.1 Introducción:

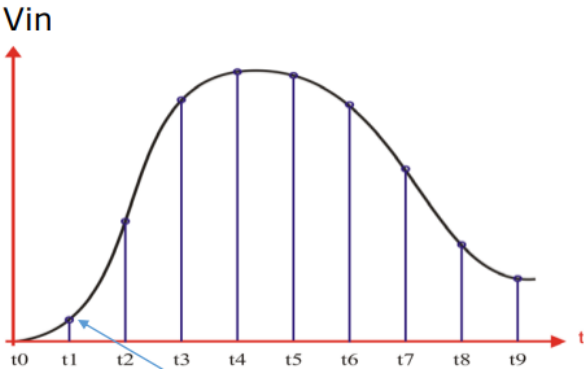
El ADC (Analog to Digital Converter) es un módulo que se utiliza para convertir una señal analógica en una señal digital. Este periférico es de suma importancia ya que, un gran porcentaje de las señales del mundo real son analógicas, es decir, que varían de forma continua en el tiempo pudiendo tomar cualquier valor dentro de un rango (ver representación 1).

Representación 1 señal analógica

Esta señal no sería posible de procesar en un MCU ya que este es un dispositivo digital, opera con señales discretas que toman valores específicos.

3.2 Funcionamiento:

Para que el MCU pueda operar con señal que esta fuera de su alcance de procesamiento, se utiliza el ADC el cual convierte tal señal en una secuencia de valores digitales. Tales valores representan la amplitud de la señal en ese tiempo específico. (ver representación 2)



Representación 2 secuencia de valores digitales

Donde t1, t2, t3, etc. son valores de la señal analógica en ese instante de tiempo.

A partir de esta conversión, el MCU puede realizar cálculos, aplicar algoritmos, tomar decisiones lógicas, etc. dependiendo de este valor instantáneo pudiendo así, interactuar con el mundo analógico.

Para realizar la conversión, utiliza un valor de voltaje de referencia que se utiliza para determinar el rango máximo de voltajes que el ADC puede medir. Cuando el ADC convierte una señal analógica a digital, compara el valor de voltaje de entrada (*Ver ecuación 3*) con este valor de voltaje de referencia. En este MCU, hay 3 tipos de referencia, en este proyecto, se utilizará la referencia de voltaje que está en el pin AVCC de 5V

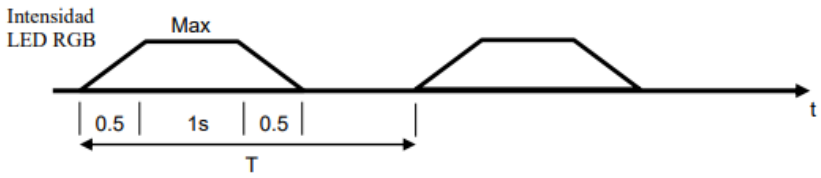
$$ADC = \frac{V_{in}.1024}{V_{ref}} \quad \textit{Ecuación 3}$$

ADC es el registro que contiene el valor luego de realizar la conversión, este registro es el que se lee para realizar acciones en base al valor.

3.3 Implementación:

Las principales características del ADC en el Atmega328P son que, su resolución es de 10 bits, es decir puede representar 2^{10}=1024 niveles discretos. Posee 8 canales de entrada analógica, del ADC0 a ADC7, pudiendo medir hasta 8 señales digitales.

En este proyecto, se utilizará el ADC obteniendo el valor que nos provee la resistencia LDR para cambiar el periodo de parpadeo del LED, estando en luz ambiente o en oscuridad (ver representación 4)



Representación 4 parpadeo del LED

Se debe, por lo tanto, configurar el reloj y tipo de conversión, configurar el puerto de entrada como analógica, seleccionar la tensión de referencia y luego, empezar la conversión.

Para configurar el ADC, se utilizan los registros de programación, entre ellos, ADMUX y ADCSRA.

3.3.1 Configuración:

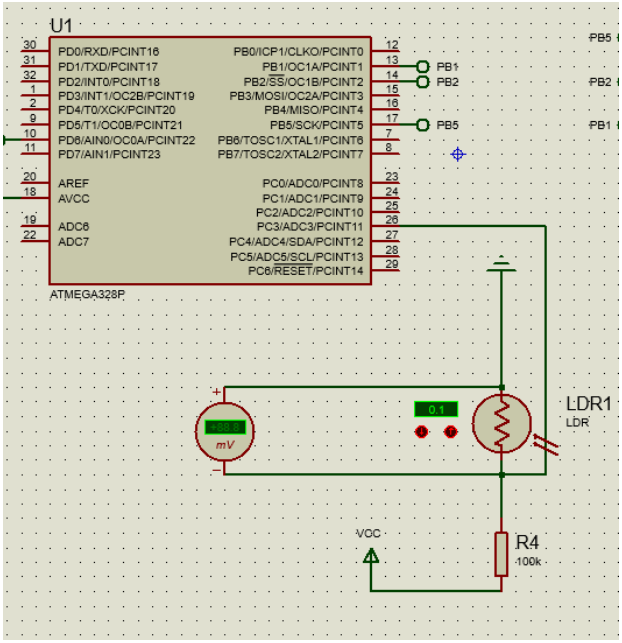
Registro ADMUX:

ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bits 3:0 – MUX3:0: Bits de selección de canal analógico

Estos bits se utilizan para determinar que pines se utilizaran como entrada analógica, en este caso, el ADC3 que le corresponde el pin C3 el cual la resistencia LDR está conectada (ver conexionado 3). En C: `ADMUX |= (1<<MUX1)|(1<<MUX0)`



Conexionado 3 LDR ADC3

Bits 5 – ADLAR: ADC Left Adjust Result

El ADLAR bit afecta a la presentación del resultado de la conversión que se deposita en el registro ADC de 16 bits. Un 1 en este bit ajustara a la izquierda el resultado, y un 0 a la derecha (ver representación 2)

ADLAR = 0

Bit	15	14	13	12	11	10	9	8	
(0x79)	–	–	–	–	–	–	ADC9	ADC8	ADCH
(0x78)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0	ADCL
	7	6	5	4	3	2	1	0	

ADLAR = 1

Bit	15	14	13	12	11	10	9	8	
(0x79)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADCH
(0x78)	ADC1	ADC0	–	–	–	–	–	–	ADCL
	7	6	5	4	3	2	1	0	

Representación 2 ajustes de ADC

Uno de las aplicaciones de ajustar a izquierda es para solo leer la parte alta del resultado, obviando los 2 valores más bajos del resultado entero en caso de que no se necesita una precisión muy alta o estos valores no sean significativos para el problema. En este caso se ajustará a derecha y utilizaremos los 1024 valores. En C: `ADMUX &=~(1<<ADLAR)`

Bits 7:6 – REFS: Bits de selección de referencia

Estos bits seleccionan el voltaje de referencia para el ADC (ver tabla 1)

REFS1	REFS0	Voltage Reference Selection
0	0	AREF, internal V_{REF} turned off
0	1	AV_{CC} with external capacitor at AREF pin
1	0	Reserved
1	1	Internal 1.1V voltage reference with external capacitor at AREF pin

Tabla 1 voltajes de referencia

Como se mencionó, se utilizará AVcc como referencia. En C: `ADMUX |= (1<<REFS0)`

Registro ADCSRA:

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit 7 – ADEN: ADC Enable

Escribir un 1 en este bit habilita el ADC, un 0 al contrario, lo apaga.

Bit 6 – ADSC: ADC Start Conversion

En modo conversión única, escribir en uno en este bit comenzara una nueva conversión

En este entregable se usará el modo conversión única, existen 2 modos más que no serán utilizados. Este bit es modificado luego de realizar toda la inicialización y se estará modificando constantemente en una loop infinito para ir realizando muchas conversiones, consultando el valor en cada una de estas.

Bit 2:0 – ADPS2:0: ADC Prescaler Select Bits

Estos bits determinan el prescalador del reloj ADC.

Se garantiza la mejor performance hasta un ADCLK de 20 KHz, por lo que aplicamos un divisor de 128, teniendo:

$$ADC_{clk} = \frac{16Mhz}{128} = 125KHZ$$

En C: `ADCSRA |= (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)`

De esta forma, tenemos el ADC del canal 3 inicializado y configurado, listo para realizar una conversión analógica digital. (ver Pseudocodigo 1)

```
Inicializar ADC
Habilitar conversión
```

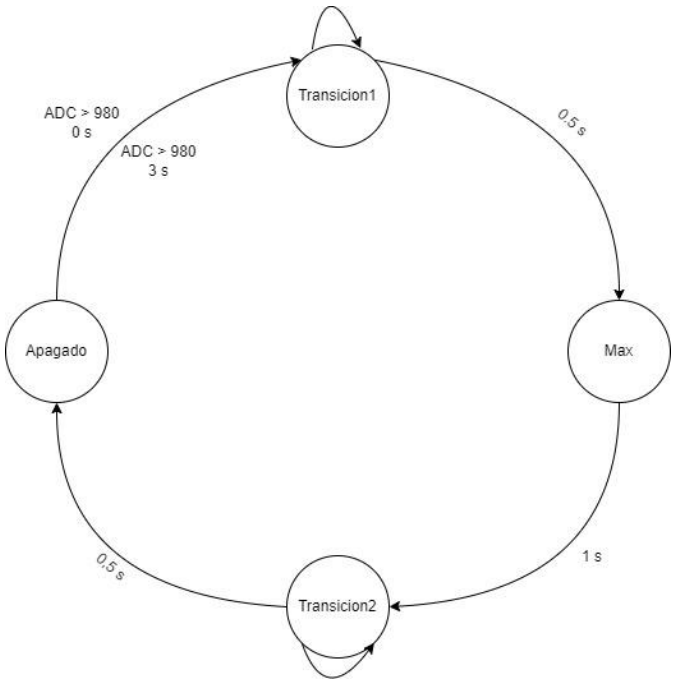
```
Loop infinito
   Mientras que la conversión no finalice, esperar //La conversión es muy rápida
   Limpiar flag ADIF
   Obtener el resultado
   Cambiar el periodo del parpadeo dependiendo del resultado
```

Pseudocodigo 1

4. MEF:

4.1 Introducción:

Para realizar la solución al periodo de parpadeo, se optó por una máquina de estados finita de Moore (ver MEF)



MEF transición entre estados

El cual cada 8 ms, una función comprobara el estado actual de la MEF y actuara dependiendo en cual este. Esta MEF esta temporizada por la rutina de interrupción del TIMER 0 el cual luego de 1000 interrupciones, levantara un flag de actualización y una función, responsable de leer los estados de los flag, actualizara la MEF y limpiara el flag.

Como se mencionó anteriormente, la transición depende del valor que provee la resistencia LDR al ADC, leyendo el registro para verificar.

4.2 Implementación:

Se generan 4 estados el cual, donde cada vez que se actualiza, se pregunta por el valor del ADC, si este es mayor a 1000 (luz ambiente) se indica en un variable colocando un 1, caso contrario (oscuridad) la variable está en 0.

A continuación, se presenta Pseudocodigo de cada estado

Apagado:

```
Si está en luz ambiente
   Estar apagado 0 segundos
   Luego de 0 segundos, transicionar a 'Transicion1'
Si no
   Estar apagado 3 segundo
   Luego de 3 segundo, transicionar a 'Transicion1'
```

Transicion1:

Transicion1 es el estado el cual el led estará en camino a prenderse al color seleccionado, es decir, aumentar la intensidad del color. Este porcentaje es tal que, en un intervalo de 0,5 segundos, ya alcanzó su máxima intensidad

```
Aumentar un porcentaje la intensidad del color
Si paso 0,5 segundos
   Transicionar a 'Max'
```

Max:

```
Si está en luz ambiente
   Estar apagado 0.5 segundo
   Luego de 0.5 segundo, transicionar a 'Transicion2'
Si no
   Estar apagado 0.5 segundo
   Luego de 0.5 segundo, transicionar a 'Transicion2'
```

Transición 2:

Transición 2 es el estado el cual el led comenzara a apagarse, bajando la intensidad de los colores un porcentaje, este porcentaje es tal que, en un intervalo de 0,5 segundos, ya se apagó completamente.

```
Disminuir un porcentaje la intensidad del color
Si paso 0,5 segundos
   Transicionar a 'Apagado'
```

Para disminuir o aumentar el porcentaje de la intensidad de color, se crea una función el cual recibe como parámetro la dirección, es decir, si debe aumentar la intensidad o disminuirla. Esta función por cada vez que se la ejecuta, modifica los registros OC1A y OC1B hasta llegar al valor esperado. (ver Pseudocodigo 2)

```
Si la dirección indica aumentar
  Si el valor del OC1A es mayor al color azul definido
    Disminuir el valor de 0C1A en 5 //Aproximadamente 2% de intensidad
  Si el valor de OC1B es mayor al color verde definido
    Disminuir el valor de 0C1B
  Si el valor de pwm_value es mayor al rojo definido
    Aumentar el valor de pwm_value en 5
Si la dirección indica disminuir
  Si el valor de OC1A es menor a 255
    Aumentar el valor de OC1A en 5
  Si el valor de OC1B es menor a 255
    Aumentar el valor de OC1B en 5
  Si el valor de pwm_value es mayor a 1
    Disminuir esta variable en 5
```

Pseudocodigo 2

La varible pwm_value es la que se describió anteriormente, la cual es PWM_DELTA. Al aumentar o disminuir esta, el ciclo de trabajo de PWM por software cambia.

5. **Conclusión**:

Al realizar este proyecto, el cual se aplican varios conceptos nuevos, como PWM, se concluye que es de gran utilidad para regular la intensidad de la luz de forma eficiente y simple. Por otro lado, tenemos el ADC de cual es de mucha importancia por el hecho de que un gran porcentaje de señales del mundo real son analógicas, permitiendo convertir una amplia cantidad de variables físicas a variables lógicas pudiendo el MCU procesar este tipo de información y generar decisiones en base al valor dado. En este caso se aprovechó la señal que genera la resistencia LDR para obtener información sobre la luminosidad del entorno.

Videos funcionamiento LED y RES LDR

https://drive.google.com/drive/folders/1Zu3yMrNYoNZGUaEsovQ1T5jtf7s1WUvn?usp=sharing