



C3.7 Programación Microcontrolador NodeMCU ESP32

Arduino y modulador de ancho de pulso PWM con salida analógica



Instrucciones

- De acuerdo con la información presentada por el asesor referente al tema, desarrollar lo que se indica dentro del apartado siguiente.
- Toda actividad o reto se deberá realizar utilizando el estilo **Markdown con extension .md** y el entorno de desarrollo VSCode, debiendo ser elaborado como un documento **single page**, es decir si el documento cuanta con imágenes, enlaces o cualquier documento externo debe ser accedido desde etiquetas y enlaces.
- Es requisito que el archivo .md contenga una etiqueta del enlace al repositorio de su documento en Github, por ejemplo **Enlace a mi GitHub**
- Al concluir el reto el reto se deberá subir a github el archivo .md creado.
- Desde el archivo .md se debe exportar un archivo .pdf con la nomenclatura **C3.7_NombreAlumno_Equipo.pdf**, el cual deberá subirse a classroom dentro de su apartado correspondiente, para que sirva como evidencia de su entrega; siendo esta plataforma **oficial** aquí se recibirá la calificación de su actividad por individual.
- Considerando que el archivo .pdf, fue obtenido desde archivo .md, ambos deben ser idénticos y mostrar el mismo contenido.
- Su repositorio ademas de que debe contar con un archivo **readme.md** dentro de su directorio raíz, con la información como datos del estudiante, equipo de trabajo, materia, carrera, datos del asesor, e incluso logotipo o imágenes, debe tener un apartado de contenidos o indice, los cuales realmente son ligas o **enlaces a sus documentos .md**, *evite utilizar texto* para indicar enlaces internos o externo.
- Se propone una estructura tal como esta indicada abajo, sin embargo puede utilizarse cualquier otra que le apoye para organizar su repositorio.

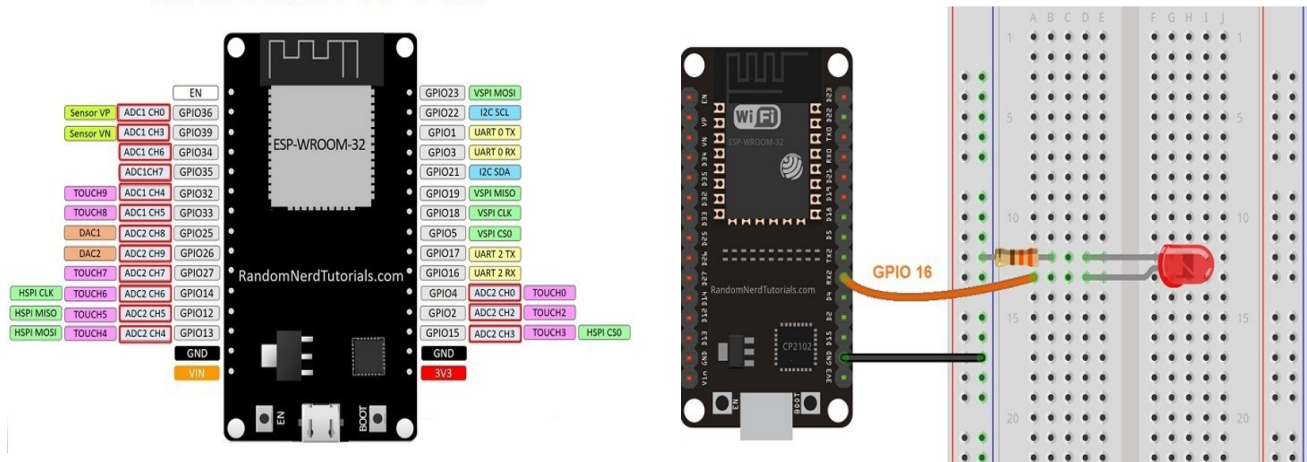
```
| readme.md
| | blog
| | | C3.1_TituloActividad.md
| | | C3.2_TituloActividad.md
| | | C3.3_TituloActividad.md
| | | C3.4_TituloActividad.md
| | | C3.5_TituloActividad.md
| | | C3.6_TituloActividad.md
| | | C3.7_TituloActividad.md
| | | C3.8_TituloActividad.md
| | img
| | docs
| | | A3.1_TituloActividad.md
| | | A3.2_TituloActividad.md
```



Desarrollo

1. Ensamble el circuito mostrado en la figura siguiente.

ESP32 DEVKIT V1 - DOIT



2. Analice y escriba el programa que se muestra a continuación.

```
// the number of the LED pin
const int ledPin = 16; // 16 corresponds to GPIO16

// setting PWM properties
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

void setup(){
  // configure LED PWM functionalities
  ledcSetup(ledChannel, freq, resolution);

  // attach the channel to the GPIO to be controlled
  ledcAttachPin(ledPin, ledChannel);
}

void loop(){
  // increase the LED brightness
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }

  // decrease the LED brightness
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    // changing the LED brightness with PWM
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }
}
```

Then, you set the PWM signal properties. You define a frequency of 5000 Hz, choose channel 0 to generate the signal, and set a resolution of 8 bits. You can choose other properties, different than these, to generate different PWM signals.

In the `setup()`, you need to configure LED PWM with the properties you've defined earlier by using the `ledcSetup()` function that accepts as arguments, the `ledChannel`, the frequency, and the resolution, as follows:

In the loop, you'll vary the duty cycle between 0 and 255 to increase the LED brightness.

And then, between 255 and 0 to decrease the brightness.

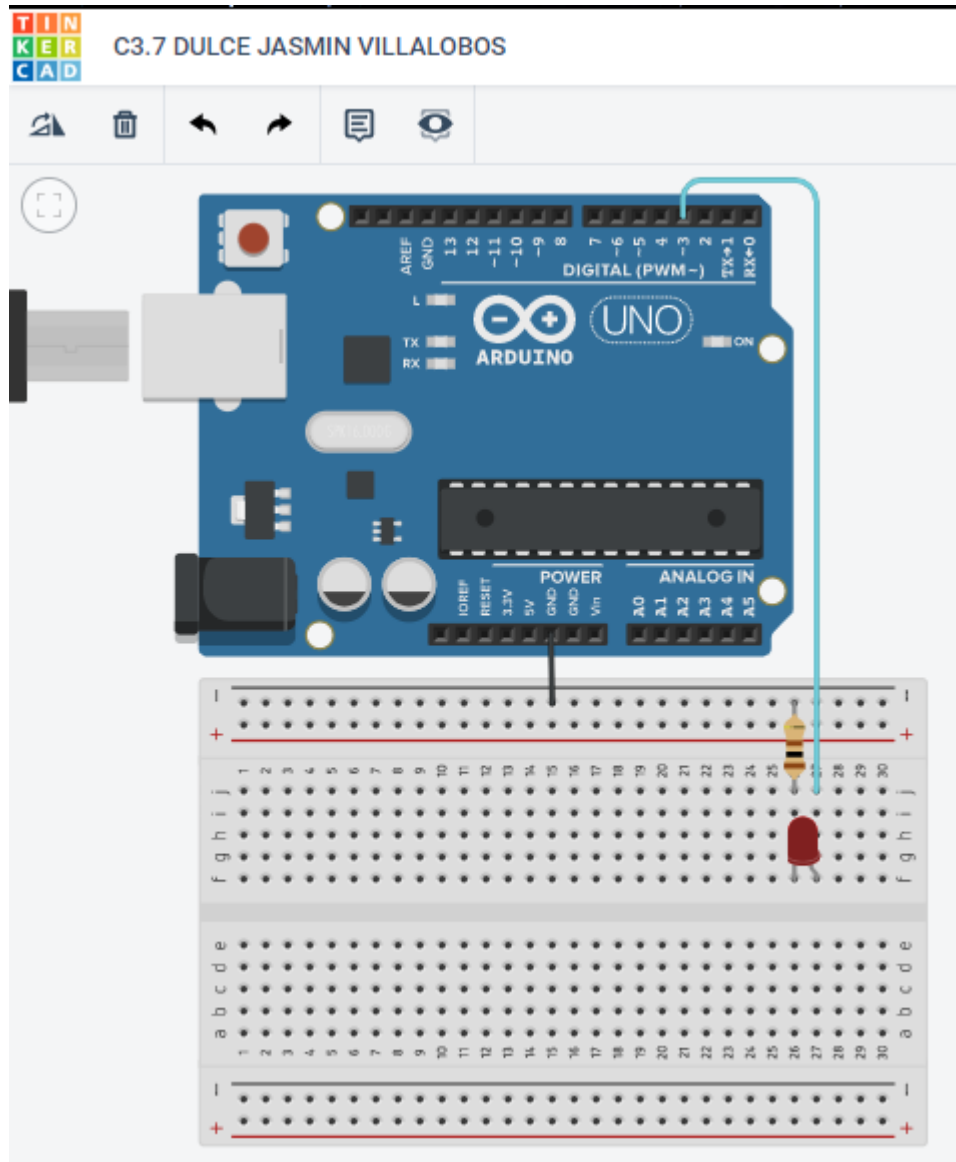
To set the brightness of the LED, you just need to use the `ledcWrite()` function that accepts as arguments the channel that is generating the signal, and the duty cycle.

Fuente de consulta: [Random Nerd Tutorials](https://randomnerdtutorials.com/esp32-led-pwm/)

3. Inserte aquí las imágenes que considere como evidencias para demostrar el resultado obtenido.

Evidencia usando el simulador.

Para generar señales PWM con un arduino, vi un tutorial para tomarlo como referencia y poder aplicarlo en un arduino en el simulador.




Para la simulación, se debía encender el led al conectar solo el puerto GPI16 a una pata del Led mientras la otra con una resistencia a Tierra GND.

```

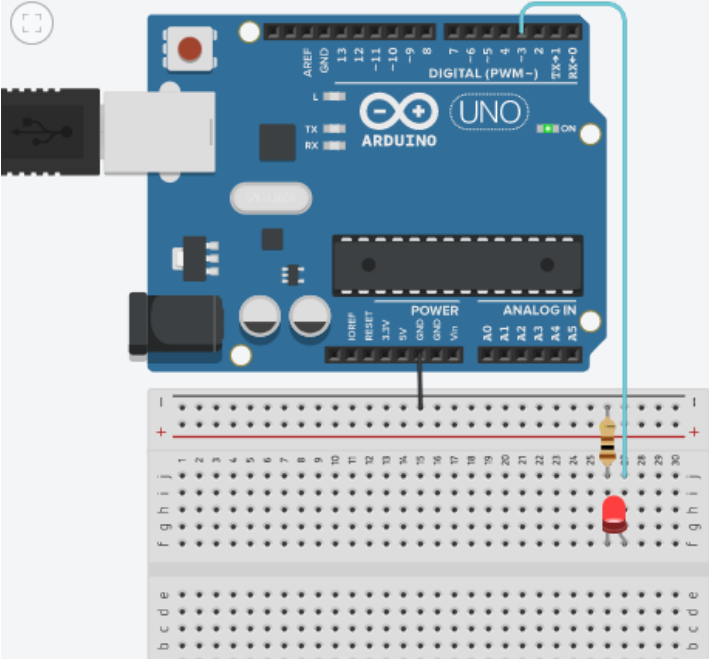
1
2 //Declaramos las variables a usar
3 const int led=3; // Led Conectado en pin 3|
4
5 int brillo;
6
7 void setup()
8 {
9   //El pin Led se declara de tipo salida
10  pinMode(led, OUTPUT);
11  // Declaramos
12 }
13
14 void loop()
15 {
16   //Aggregamos dos ciclos for para
17   //la representacion del los valores dentro
18   //del pontenciometro
19   for(brillo = 0; brillo < 256; brillo++)
20   { // El brillo del led varia
21     analogWrite(led,brillo); delay(10);
22   }
23
24   delay(1000);
25   //El segundo ciclo es para darle continuidad
26   //de forma inversa.
27   for(brillo = 255; brillo >= 0; brillo --)
28   {
29     analogWrite(led,brillo); delay(10);
30   }
31   delay(1000);
32 }
33

```

Para simular eso se pone en el codigo los dos for, como en el ejemplo, de forma que usara los valores del pontenciometro, y asi el led se fuera apagando y encendiendo.


C3.7 DULCE JASMIN VILLALOBOS

Hora de simulador: 00:00:03



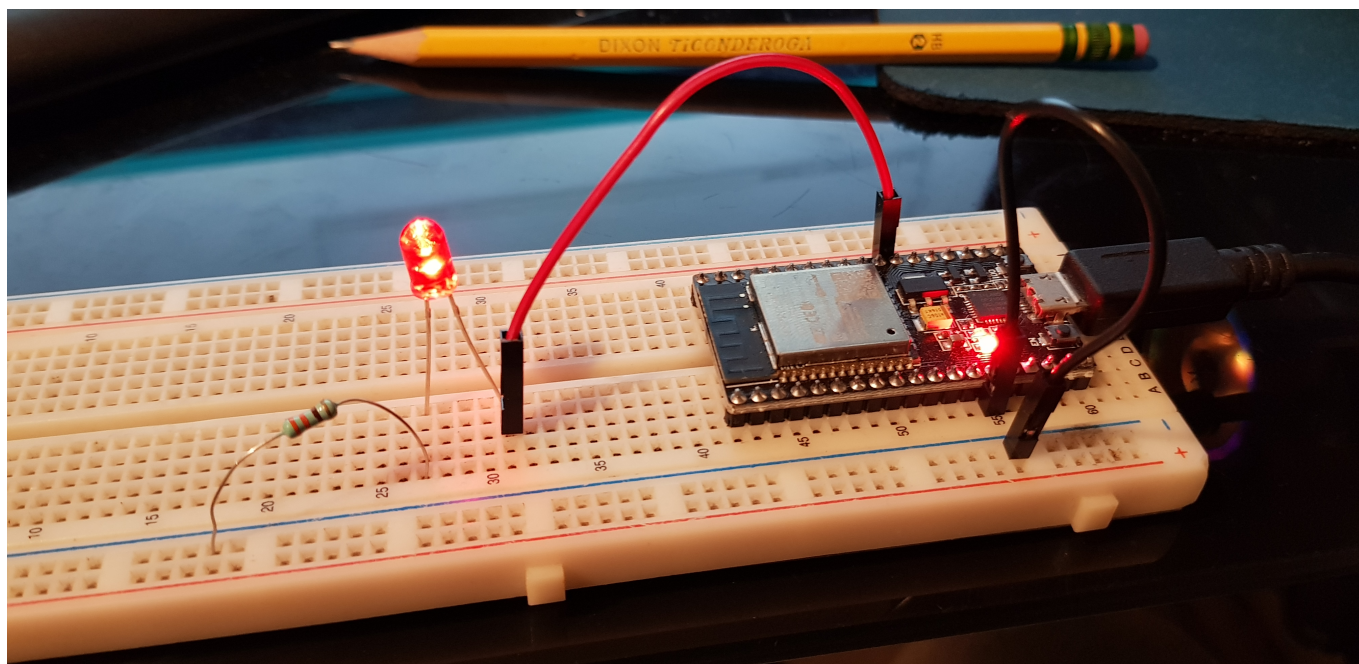
Texto

```

1
2 //Declaramos las variables a usar
3 const int led=3; // Led Conectado en pin 3
4
5 int brillo;
6
7 void setup()
8 {
9   //El pin Led se declara de tipo salida
10  pinMode(led, OUTPUT);
11  // Declaramos
12 }
13
14 void loop()
15 {
16   //Aggregamos dos ciclos for para
17   //la representacion del los valores dentro
18   //del pontenciometro
19   for(brillo = 0; brillo < 256; brillo++)
20   { // El brillo del led varia
21     analogWrite(led,brillo); delay(10);
22   }
23
24   delay(1000);
25   //El segundo ciclo es para darle continuidad
26   //de forma inversa.
27   for(brillo = 255; brillo >= 0; brillo --)
28   {
29     analogWrite(led,brillo); delay(10);
30   }
31   delay(1000);
32 }
33

```

Evidencia proporcionada por el compañero con los componentes:



C3.7 Arduino 1.8.12

Archivo Editar Programa Herramientas Ayuda



C3.7 §

```
//Seleccionamos que el GPIO16 sera la que usaremos
const int ledPin = 16;

//Propiedades de PWM
const int freq = 5000;
const int ledChannel = 0;
const int resolution = 8;

void setup() {
  //Se configura el comportamiento del LED a una frecuencia de 5000
  ledcSetup(ledChannel, freq, resolution);
  //Aqui se declara el ledPin para ser controlado
  ledcAttachPin(ledPin, ledChannel);
}

void loop() {
  //Incrementa el brillo del LED cuando el valor de dutyCycle es igual o menor a 255
  for(int dutyCycle = 0; dutyCycle <= 255; dutyCycle++){
    //se cambia el brillo del LED segun el valor de dutyCycle
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }

  //Disminuye el brillo del LED cuando el valor de dutyCycle es 255
  for(int dutyCycle = 255; dutyCycle >= 0; dutyCycle--){
    //se cambia el brillo del LED segun el valor de dutyCycle
    ledcWrite(ledChannel, dutyCycle);
    delay(15);
  }
}
```

Criterios	Descripción	Puntaje
Instrucciones	Se cumple con cada uno de los puntos indicados dentro del apartado Instrucciones?	20
Desarrollo	Se respondió a cada uno de los puntos solicitados dentro del desarrollo de la actividad?	80

[ENLACE - MI GITHUB](#)