



## Tecnológico de Monterrey

Reto semanal

**Semana 3**

**Manchester Robotics**

### **Integrantes:**

A017314050 | Eleazar Olivas Gaspar

A01735696 | Azul Nahomi Machorro Arreola

A01732584 | Angel Estrada Centeno

A01735692 | Arick Morelos del Campo

### **Profesores:**

Rigoberto Cerino Jimenez

Alfredo Garcia Suarez

Juan Manuel Ahuactzin Larios

Marzo 2024

### **Resumen:**

Tendremos un acercamiento con dispositivos programables y elementos electrónicos básicos en el área de la robótica, como potenciómetros, LEDs, entre otros.

### **Objetivos:**

Desarrollar capacidades de manejo del framework ROS, e incrementar el grado de interacción y la aplicación de funciones en ROS. Este reto semanal nos permitirá visualizar la respuesta de estos sistemas ante ciertos estímulos programados, realizando una sinergia entre hardware y el sistema de ROS.

### **Introducción:**

Micro-ROS (micro Robot Operating System) es una implementación de ROS 2 (Robot Operating System 2) diseñada para sistemas embebidos y microcontroladores. ROS es una plataforma de código abierto utilizada comúnmente en robótica para el desarrollo de software. Micro-ROS se adapta a entornos con recursos limitados, como microcontroladores, para facilitar la integración de sistemas robóticos más pequeños y eficientes. Micro-ROS proporciona un conjunto reducido de características de ROS 2 para adaptarse a los recursos limitados de microcontroladores. Utiliza el middleware de comunicación DDS (Data Distribution Service) para permitir la comunicación entre los diferentes nodos de ROS en un sistema robótico. Además, Micro-ROS ofrece una integración estrecha con herramientas y bibliotecas estándar de ROS, permitiendo a los desarrolladores trabajar en un entorno familiar incluso cuando se trabaja en sistemas embebidos.

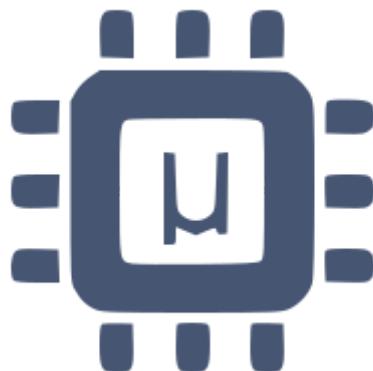


Fig. 1. logo de Micro Ros

Un ADC es un componente electrónico que convierte señales analógicas continuas en valores digitales discretos. En el contexto de microcontroladores como el ESP32, los módulos ADC permiten medir señales analógicas provenientes de sensores u otros dispositivos analógicos. El proceso de conversión comienza cuando el ADC muestrea la señal analógica a intervalos regulares. Luego, cuantifica la amplitud de cada muestra y la representa como un valor digital. Los microcontroladores, como el ESP32, suelen tener múltiples canales ADC que permiten la conexión de varios sensores analógicos. Los valores digitales resultantes pueden ser utilizados por el microcontrolador para tomar decisiones o ser transmitidos a través de una red para su procesamiento adicional.

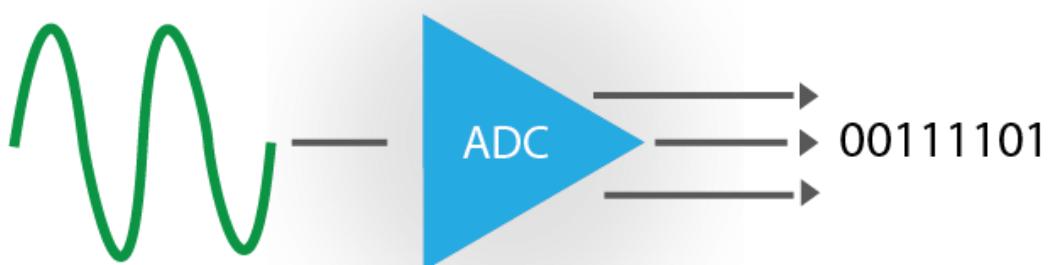


Fig. 2. Convertidor ADC

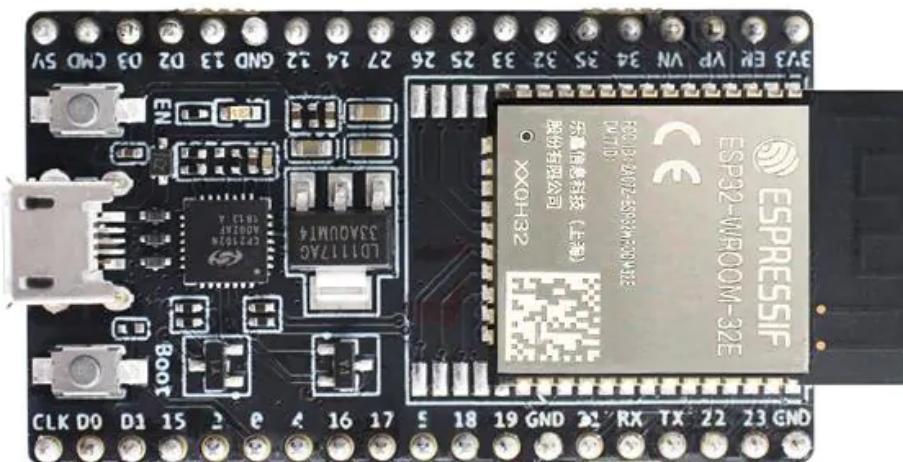


Fig. 3. Esp32 a utilizar

PWM es una técnica que se utiliza para modular la amplitud de una señal de onda cuadrada, conocida como pulso, con el fin de controlar la cantidad de energía entregada a un dispositivo. En microcontroladores como el ESP32, los módulos PWM se utilizan para

controlar la velocidad de motores, el brillo de LEDs y otras aplicaciones donde se requiere control de potencia. El módulo PWM del ESP32 genera una señal de onda cuadrada con un ancho de pulso variable. La relación entre el tiempo que la señal está en alto (on) y el período total de la señal determina la potencia entregada al dispositivo conectado al pin PWM. Al variar este ancho de pulso, se puede lograr un control preciso de la velocidad, intensidad luminosa, etc. Los microcontroladores suelen tener varios pines con capacidad PWM para ofrecer flexibilidad en el control de diferentes dispositivos.

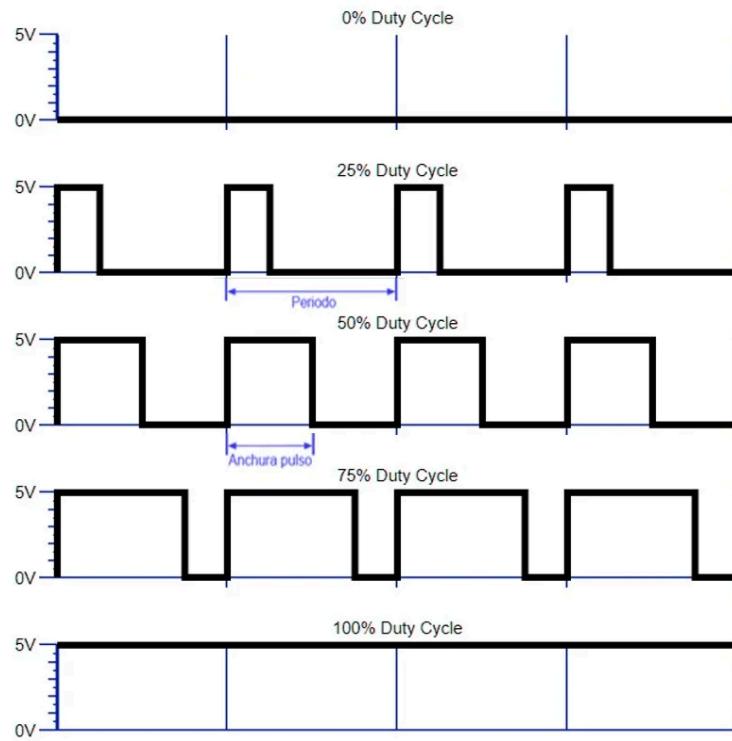


Fig. 4. Gráficas de los ciclos de un PWM

## Solución del problema:

-Declaración de variables a usar

```
12 //Publishers
13 rcl_publisher_t publisher_raw_pot;
14 rcl_publisher_t publisher_voltage;
15
16 //Subscriber
17 rcl_subscription_t direcrion_motor, duty_cycle_subs;
18
19 //msg type int
20 std_msgs_msg_Int32 dir;
21
22 //msg type float
23 std_msgs_msg_Float32 raw_pot, voltage, duty_cycle;
24
25 //general handling
26 rclc_executor_t executor;
27 rclc_support_t support;
28 rcl_allocator_t allocator;
29
30 //doe def
31 rcl_node_t node_handle;
32 rcl_timer_t timer10, timer100;
33
```

Fig. 5. Declaraciones en código

En las líneas 13 y 14 se definen los publishers raw\_pot para la lectura del potenciómetro y voltage para la conversión y salida de voltaje. En la línea 17 se define el subscriber para el ciclo del pwm. En las líneas 20 y 23 se definen los tipos de mensajes a utilizar: Int y Float. Las líneas 26 a 28 son para los manipuladores o handlings. Por último, las líneas 31 y 32 son para manipular el nodo y los timers respectivamente.

-Se implementan dos timers

```
53 void timer10_callback(rcl_timer_t * timer, int64_t last_call_time)
54 {
55     RCLC_UNUSED(last_call_time);
56     if (timer != NULL) {
57         pot = analogRead(POT_PIN)// Corrected to read the potentiometer
58     }
59 }
60
61 void timer100_callback2(rcl_timer_t * timer2, int64_t last_call_time)
62 {
63     RCLC_UNUSED(last_call_time);
64     if (timer2 != NULL) { // Corrected the timer variable check
65         raw_pot.data = pot;
66         voltage.data = pot*3.3/4160
67         RCSOFTCHECK(rcl_publish(&publisher_raw_pot, &raw_pot, NULL)); // Updated to publish Float32 message
68
69         RCSOFTCHECK(rcl_publish(&publisher_voltage, &voltage, NULL));
70     }
71 }
```

Fig. 6. Funciones de los timers

Funciones de los timers. El primero publica la lectura del potenciómetro en un rango de 0 a 255. El segundo timer lee la publicación del primer timer y hace la conversión a una salida de voltaje de 0 a 3.3 V como máximo, eso da la intensidad con la que se ilumina el LED.

- Circuito utilizado:

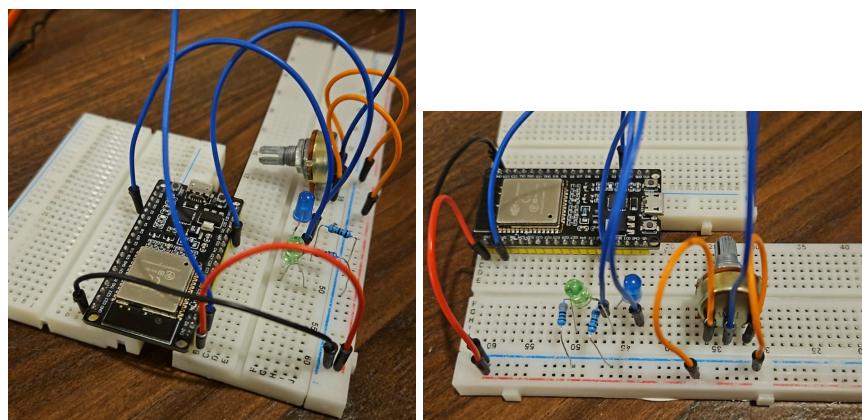


Fig. 7 y 8. Implementación física

### Resultados:

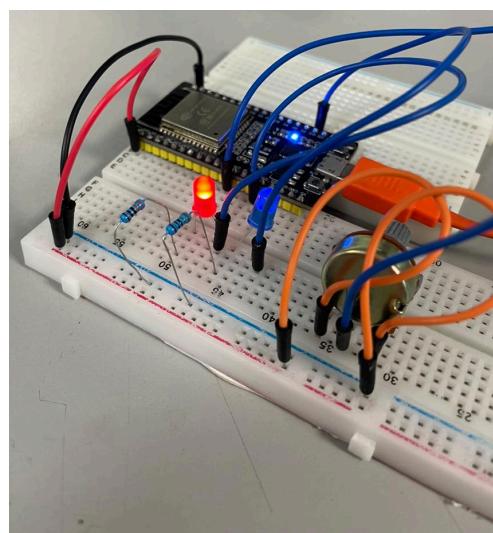


Fig. 9. Resultado final

En esta imagen se puede observar como el LED rojo varía su intensidad de luz acorde al potenciómetro. El LED azul parpadea en caso de haber algún error con el programa. La salida hacia el LED rojo es realmente la salida del PWM que irá conectada a un motor en la próxima entrega.

### **Conclusión:**

Se lograron los objetivos, en cuanto al manejo de micro ros, el potenciómetro, los leds, el pwm y la ESP32, sin embargo no integramos la parte del motor dentro de nuestra solución. Logramos hacer que mediante la comunicación de los timer se controle la intensidad del led, ya que el primer timer publica el rango de valores del potenciómetro y con este controlamos el voltaje que se le asigna al led y el que le da la intensidad.

Los retos que se presentaron fueron en función de la lectura de datos, ya que nos costó entender cómo se comunicaba nuestro código de arduino con micro ros y esto provocó que tuviéramos problemas con la solución del reto semanal, sin embargo ahora ya nos queda claro cómo es que se logra la comunicación en este tipo de programación, el hecho de que pudiéramos visualizar con base en el hardware utilizando ayudó a que pudiéramos entender más a fondo el ambiente de ros y su funcionamiento aplicado.

### **Bibliografía:**

ROS - Robot Operating System. (s. f.). <https://www.ros.org/>

ROS2 Documentation. (s.f.). <https://docs.ros.org/>

Macip. Qué es un ADC: Convertidor de analógico a digital. (s.f.).  
<https://www.masip.es/diccionario/adc/>

Documentación oficial de Espressif (fabricante de ESP32). (s.f.).  
<https://docs.espressif.com/projects/esp-idf/en/stable/esp32/>

Geeknetic. ¿Qué es y para qué sirve el PWM? (s.f.).

<https://www.geeknetic.es/PWM/que-es-y-para-que-sirve#:~:text=El%20PWM%2C%20como%20concepto%20general,un%20dispositivo%20tiene%20que%20funcionar.>