**SISTEMAS DE CONTROL EMBEBIDOS**

**PRÁCTICA 2.1**

**PWM-ADC**

1. **Objetivos**

* Configurar los parámetros de un sistema embebido para una conversión analógica digital.
* Adquirir datos analógicos hacia un sistema embebido.
* Mostrar las señales en una pantalla Oled.

1. **Recursos**

* ESP32
* Potenciómetro
* Oled 0.93”

1. **Marco teórico** 
   1. **ADC**

En el mundo moderno, la mayor parte de la información se transmite de manera digital entre dispositivos electrónicos, no obstante, gran parte de esta información es originalmente analógica. Las señales analógicas captadas mediante sensores varían constantemente y a diferentes velocidades, pasando de velocidades lentas en señales como la temperatura a velocidades rápidas como una señal de audio. El inconveniente surge cuando estas señales son muy difíciles de manipular e inclusive de guardar y recuperar con exactitud, de modo que como solución a este problema surge el ADC.



Ilustración 1 Convertidor analógico a digital

El ADC (Analog to Digital Converter) o Conversor analógico a digital es un circuito integrado que partiendo de una señal analógica puede obtener una señal digital equivalente que una máquina puede interpretar, manipular y guardar fácilmente. La exactitud con la que esta señal se convierte depende de la resolución del ADC y eso suele ser diferente en los microcontroladores.

La resolución del ADC se entiende como el voltaje necesario (señal analógica) para lograr que en la salida digital haya un cambio en el bit menos significativo. En otras palabras, el voltaje (en señal digital) obtenido por unidad de voltaje en señal analógica.

Mientras mayor sea la resolución, más preciso sería la conversión obtenida por el ADC. En el caso del ESP32, el número de bits por defecto es de 12 bits, superior a la resolución del Atmega328P (Arduino) cuyo número de bits para el ADC es de solo 8 bits, sin embargo, se puede configurar el ADC parar que trabaje con una resolución de 9 bits. Estos 12 bits por defecto del ESP32 indican que el rango de lectura de las entradas analógicas varía entre 0 y 4095, a diferencia del Atmega328P que varían del 0 al 255. Esta información es muy importante para obtener qué cantidad de voltaje estaríamos recibiendo como señal digital si recibimos X cantidad de señal analógica.

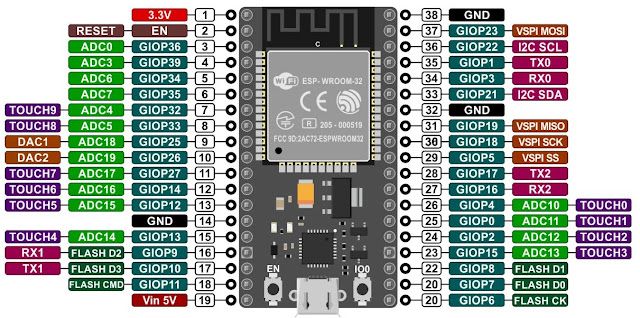
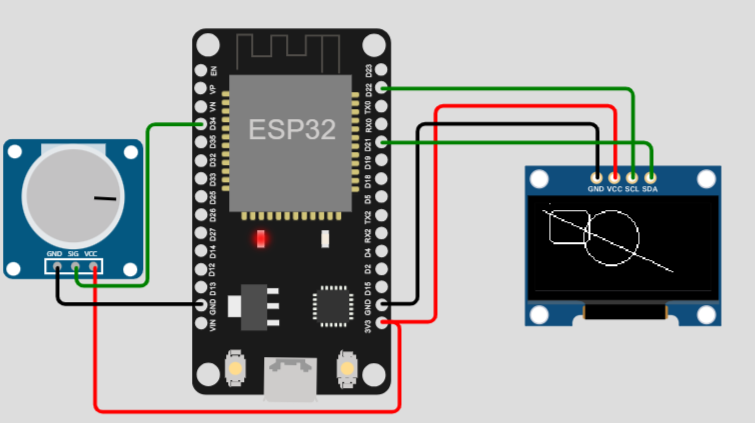


Ilustración 2 Pinout ESP32

El ESP32 cuenta con 15 pines ADC y para hacer uso de cada uno basta con inicializar el pin a utilizar como entrada y colocar este pin dentro de la función ***analogRead(pin),***esta función retorna un valor entero de 0 a 4095 (cuando se usa el ADC por defecto) que correspondería a la lectura analógica.

1. **Actividad por desarrollar**
   1. **ADC**

**Diagrama Esquemático**



**ADC:** GPIO 26

**Leds:**

GPIO 2

GPIO 4

GPIO 5

GPIO 18

GPIO 19

GPIO 21

GPIO 22

GPIO 23

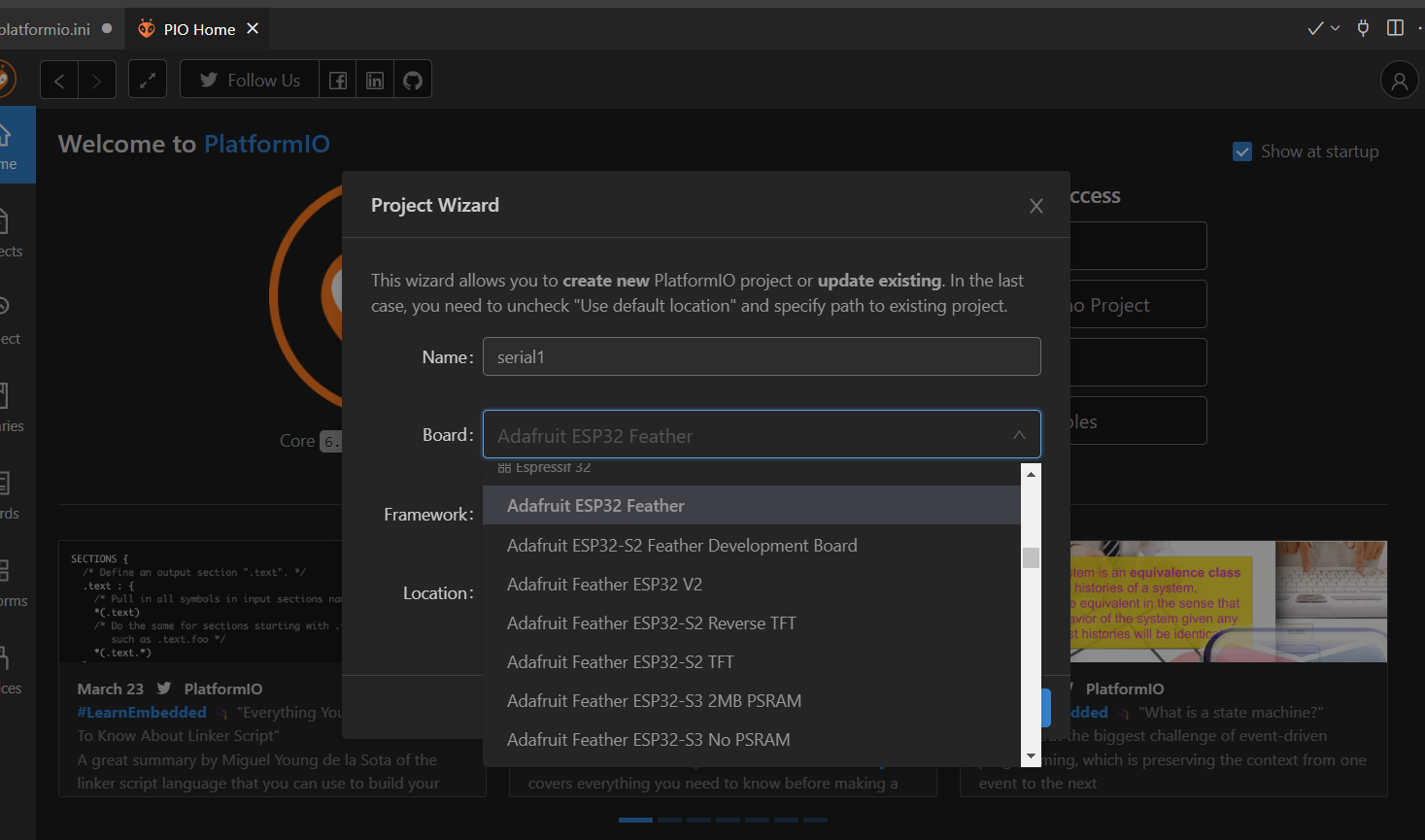
GPIO 32

GPIO 33

Ilustración 4 Diagrama esquemático para el ADC en ESP32

**Procedimiento**

1. Crear un proyecto nuevo en *platformio* y seleccionar la tarjeta *adafruit esp32 feather.*



1. Copiar el código **oled\_gráficas.cpp** del repositorio en el archivo *main.cpp* del Visual Studio Code

[https://github.com/SistemasEmbebidos2020/Oled\_Pot\_MACI/blob/main/oled\_graficas.cpp]( https://github.com/SistemasEmbebidos2020/Oled_Pot_MACI/blob/main/oled_graficas.cpp  )

1. Cargar el código y visualizar lo que asoma en la pantalla.
2. En el código se ha colocado detalladamente en cada línea de código cuál es su función

**Código utilizado**

**-oled\_gráficas.cpp**

* Se incluye la librería <Arduino.h> que permite el uso de las funciones de Arduino en general para ESP32.



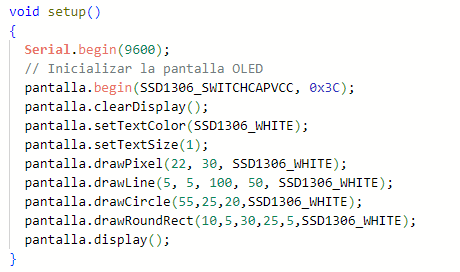
* Se declaran las librerías a utilizarse para la oled



* Se declara la variables de ancho y alto de la pantalla para especificarlas en su respectiva instancia.

Texto

Descripción generada automáticamente



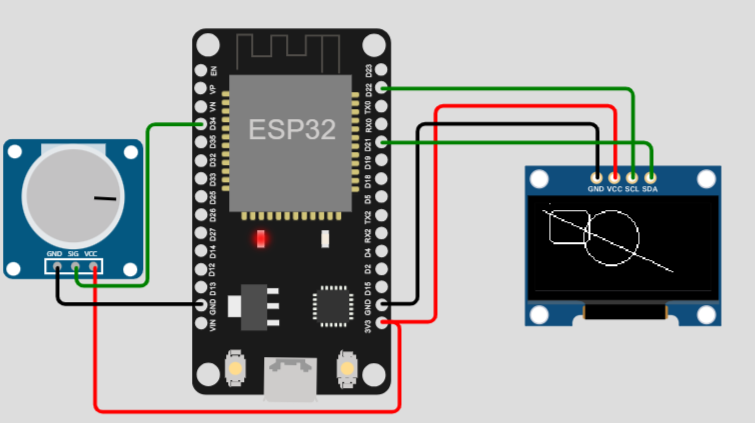
* Para el Setup, se inicializa la comunicación serial indicando los baudios(la velocidad de transmisión de datos), además se inicializa la comunicación con la pantalla oled especificando su dirección I2C .

Imagen que contiene Aplicación

Descripción generada automáticamente

* Para el loop, se deja en blanco para este primer ejemplo
  1. **Oled\_analog\_Graph**

**Diagrama esquemático**



**ADC:** GPIO 26

**Leds:**

GPIO 2

GPIO 4

GPIO 5

GPIO 18

GPIO 19

GPIO 21

GPIO 22

GPIO 23

GPIO 32

GPIO 33

**Procedimientos**

1. Reemplazar el código del main.cpp por el código del archivo Pot\_oled.cpp y cargarlo
2. <https://github.com/SistemasEmbebidos2020/Oled_Pot_MACI/blob/main/Pot_oled.cpp>
3. Visualizar el resultado moviendo el potenciómetro

**Código utilizado**

**-** **Pot\_oled.cpp**

* Se incluye la librería <Arduino.h> que permite el uso de las funciones de Arduino en general para ESP32 y permite utilizar las funciones para PWM.



* Se declaran las librerías a utilizarse para la oled



* En el setup, se inicia declarando a comunicación serial, además se inicializa la pantal oled con su respectiva dirección i2c, a continuación se hace una limpieza de la pantalla, siguiente se establece el color del texto que se mostrará en la pantalla y por ultimo pero no menos importante se especifica el tamaño del texto a escribir en la oled, finalmente se llama a la función drawejes().

Texto

Descripción generada automáticamente

* Para el la función draw ejes se establece una línea vertical y otra línea horizontal, en el cual dentro de estos “bordes” se graficará en tiempo real la lectura analógica.

Texto

Descripción generada automáticamente

* 1. **Desafío**

1. Usar un segundo potenciómetro y mostrar ambas señales superpuestas en la pantalla oled.