Workshop
"Introducción a la Robótica con Microcontroladores"

Módulo 2:
Sensores Básicos

Ing. Martín Baudino: martinbaudino@gmail.com
Ing. Pablo Garrone: pdgarrone@gmail.com

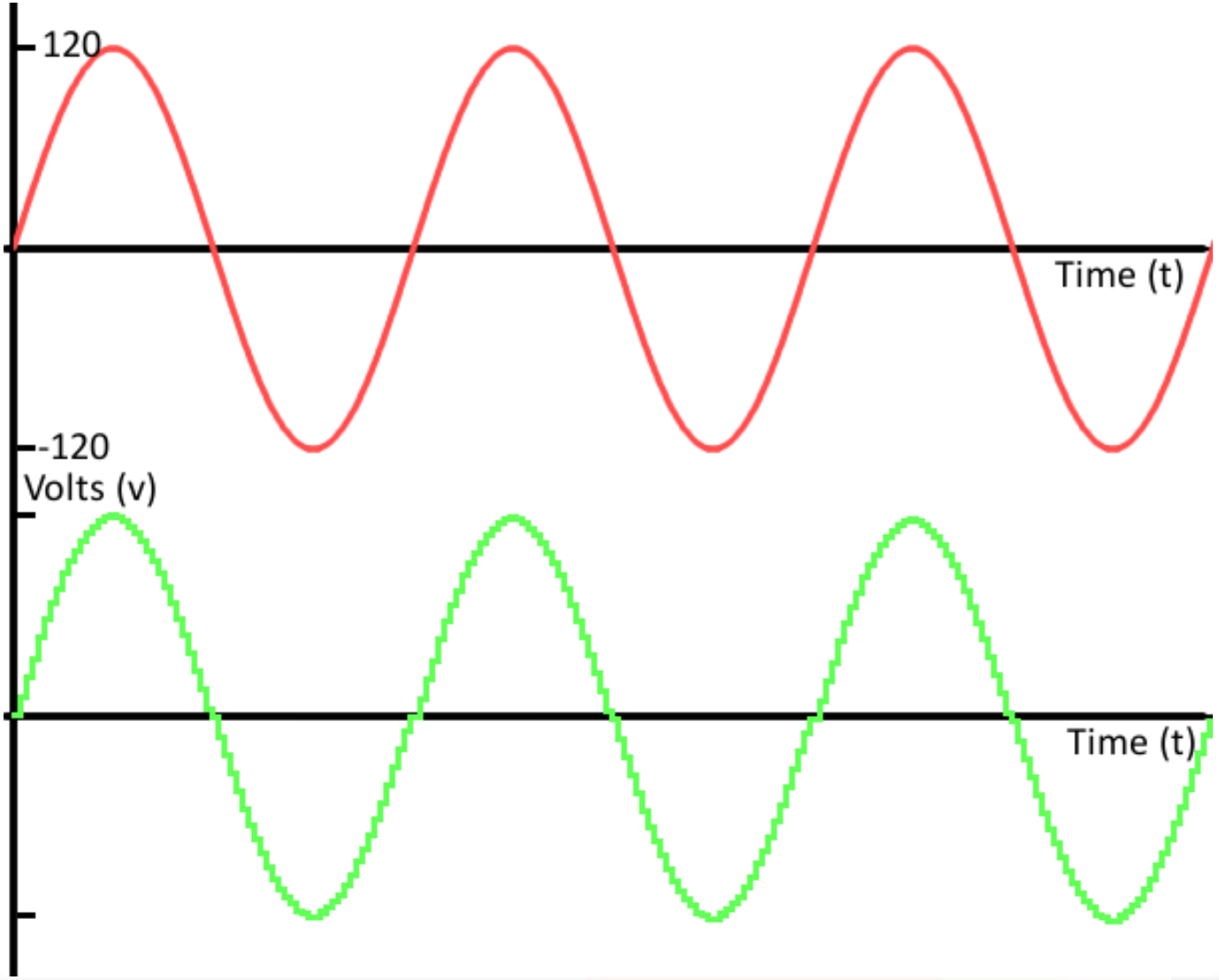
Estructura del Workshop

- Módulo 2: Sensores Básicos
  - Conversor Analógico a Digital (ADC)
  - Sensores reflectivos:
    - De luminosidad por infrarrojo
    - De distancia por ultrasonido
  - Medición y administración de tiempos
  - Temporizadores del LPC4337
  - Módulo de Captura
  - Ejercicio final: robot seguidor de líneas con detección de obstáculos

Conversión Analógica a Digital

- Microcontroladores trabajan con números discretos
- Para representar una señal analógica, primero se debe "digitalizar"
- Esto significa generar una representación discreta de la señal continua
- Para ello se utiliza un dispositivo interno, el Conversor Analógico Digital (ADC)
- Dependiendo de la cantidad de bits del ADC, será la cantidad de códigos posibles

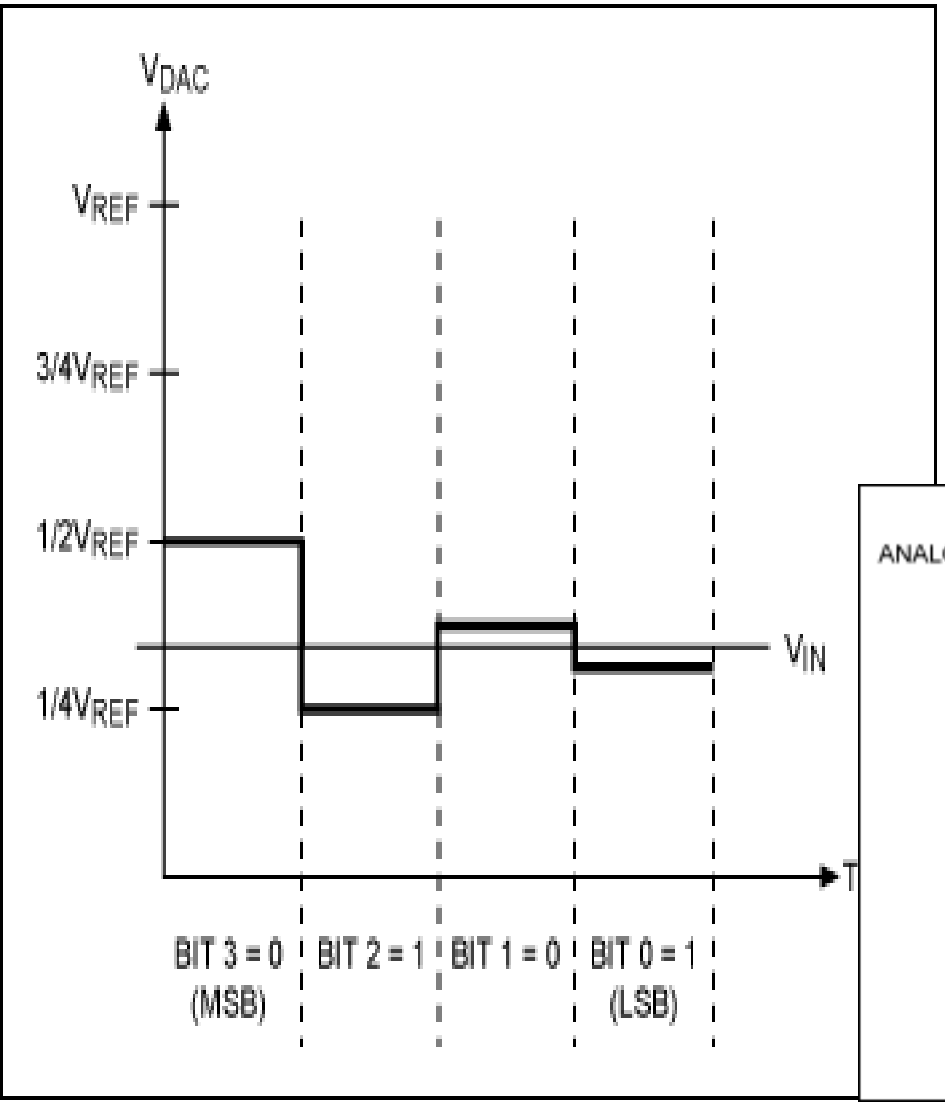
Conversión Analógica a Digital



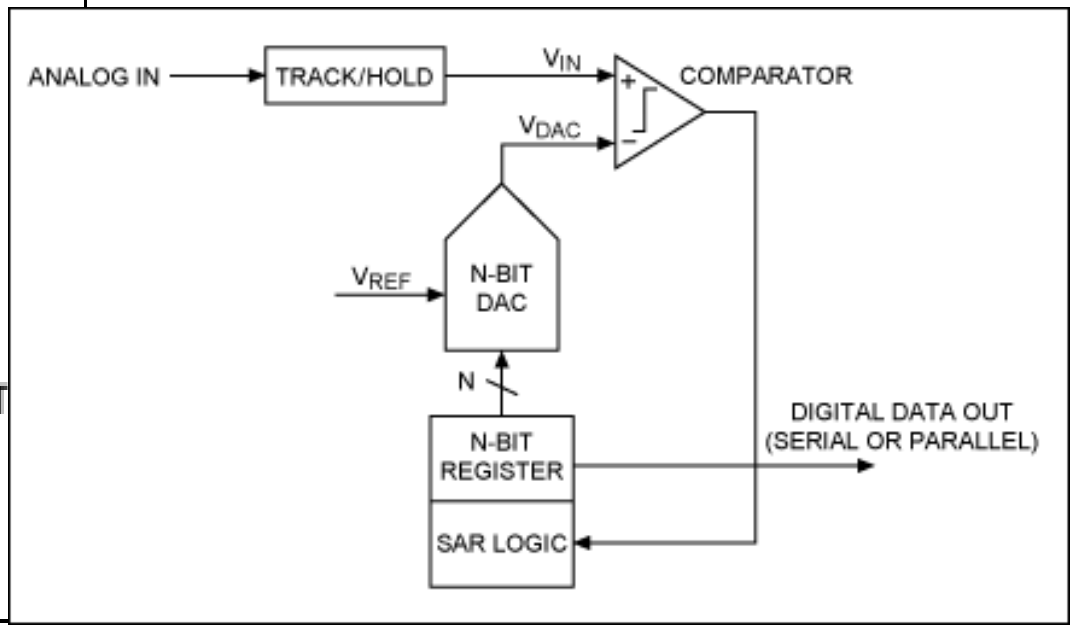
Conversión Analógica a Digital

- 2x ADCs de 10 bits de aproximaciones sucesivas
- 8x Canales de entrada por ADC
- Registros de resultados individuales por canal (<IRQs?)
- Ráfagas de conversión para una o múltiples entradas
- Rango de Medición = 0 a 3,3V
- Tiempo de conversión para 10 bits = 2,45us
- Tasa de conversión = 400 kMuestras/s
- Conexión con DMA
- Conversión en transición de entrada o Match de Temporizador

Conversión Analógica a Digital

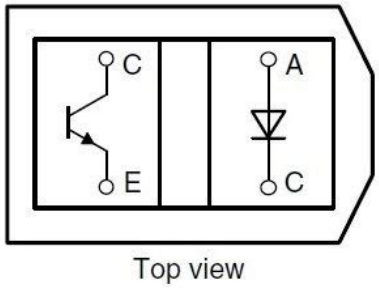


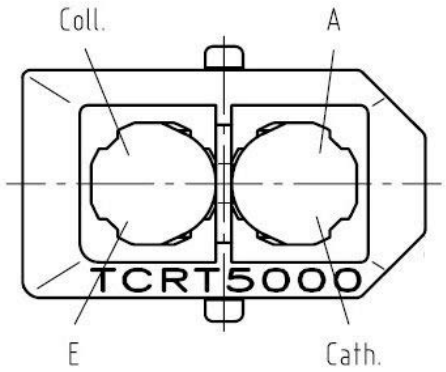
- Necesita DAC, comparador y reloj
- Tarda 11 ciclos en generar un código

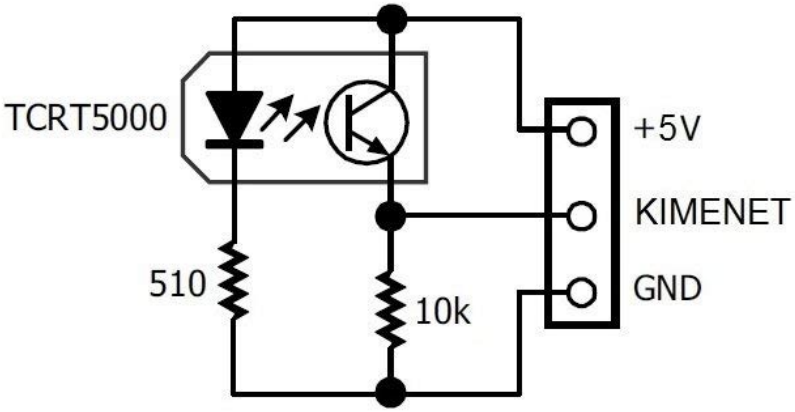


Sensor: Luminosidad por IR

- TCRT5000
  - Conectado a CH1 y CH2 de ADC







Ejercicio 4: ADC

- Leer los sensores IR y encontrar un valor que sirva como umbral entre los colores negro y blanco
  - Sensor izquierdo → adcRead(CH1);
  - Sensor derecho: → CH2
- Imprimir los valores por consola utilizando las funciones:
  - `print("Imprime cadena");`
  - `println("Imprime cadena con \r\n");`
  - `print_int(int valor);//Imprime entero`

Ejercicio 5: Seguidor de línea

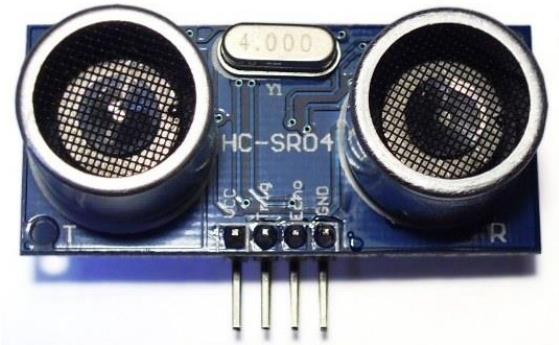
- Implementar las tareas de seguidor de línea utilizando la función ya implementada:
  - `uint8_t seguir_linea(void);`
- Devuelve los siguientes valores:
  - `LINEA_ADELANTE`
  - `LINEA_GIRAR_DERECHA`
  - `LINEA_GIRAR_IZQUIERDA`
  - `LINEA_DETENER`
- ¿Qué forma de control de flujo conviene implementar?

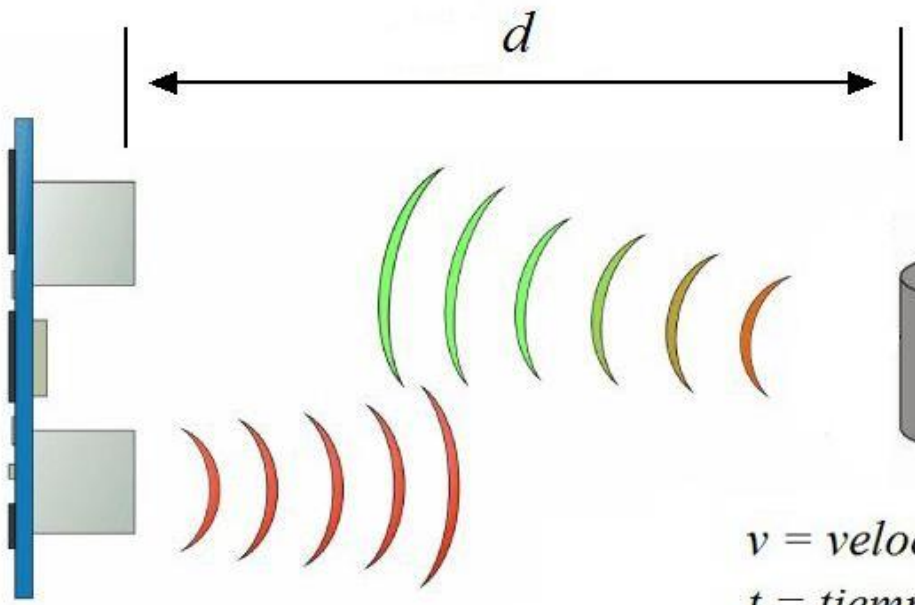
Medición de Tiempo

- Temporizadores son contadores de eventos
- Generando eventos a una cadencia fija, podemos determinar el tiempo transcurrido a partir del número de cuentas:
  - Tiempo = cnt * periodo_reloj = cnt / frec_reloj
- Microcontroladores tienen múltiples temporizadores para distintas aplicaciones:
  - Medición de tiempo transcurrido
  - Generación de eventos a intervalos fijos
  - Alarmas
  - Generación y/o decodificación de señales (ej: PWM)

Sensor: Distancia por Ultrasonido

- HCSR-04
  - Trigger: GPIO0
  - Echo: GPIO2





$$d = v * (t/2)$$

v = velocidad del sonido = 0.034 cms/us

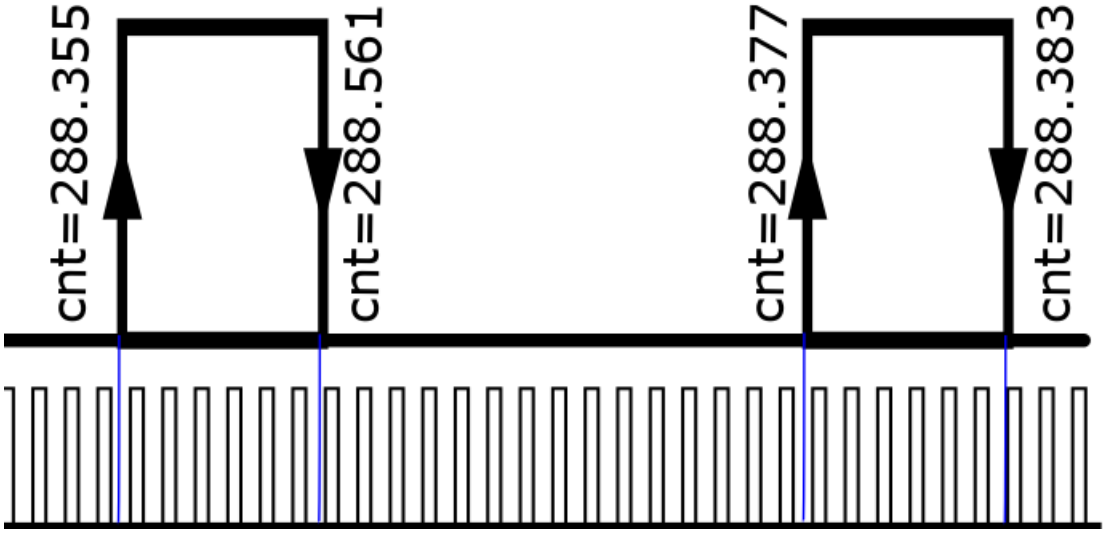
t = tiempo medido de ida y regreso

Ejercicio 6: Sonar

- Agregar al algoritmo del seguidor de líneas la capacidad de detección de obstáculos, utilizando la función de sAPI
  - `float ultrasonicSensorGetDistance( ultrasonicSensorMap_t aSensor, unitMap_t anUnit)`
- Detener el robot cuando se encuentre un obstáculo a menos de 10cm
- Al querer retomar la trayectoria ¿Se produce el mismo problema que en el ejercicio 4?

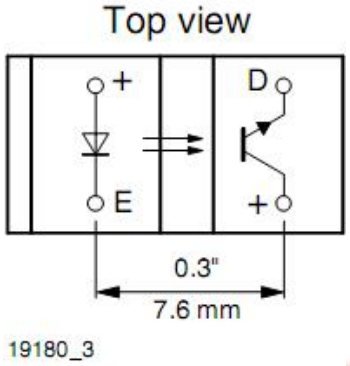
Captura de Entrada

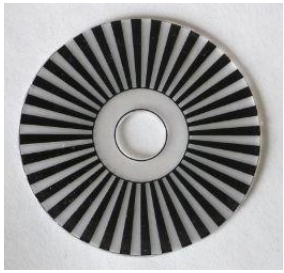
- Utilizado para decodificar señales digitales
- Cuando una señal en pin de entrada cambia de estado, guarda el número de cuentas del temporizador en un registro específico de manera automática



Encoders Ópticos

- Sensor óptico TCST2103 como el TCRT5000
- Utilizando un disco ranurado generamos pulsos digitales cuyo ancho depende de la velocidad de rotación de cada rueda
- Con módulo de captura se pueden contar la cantidad de pulso y medir el ancho





Ejercicio 7: Integración

- Agregar odometría al seguidor de líneas, leyendo las señales de los encoders con el módulo de captura de entrada
- Determinar velocidad de giro de cada rueda y distancia recorrida, utilizando la función:
  - `void get_encoders(encoder_t *izq, encoder_t *der)`
- ¿Qué particularidad tiene esta función?
- ¿Qué dato falta para saber la distancia recorrida?