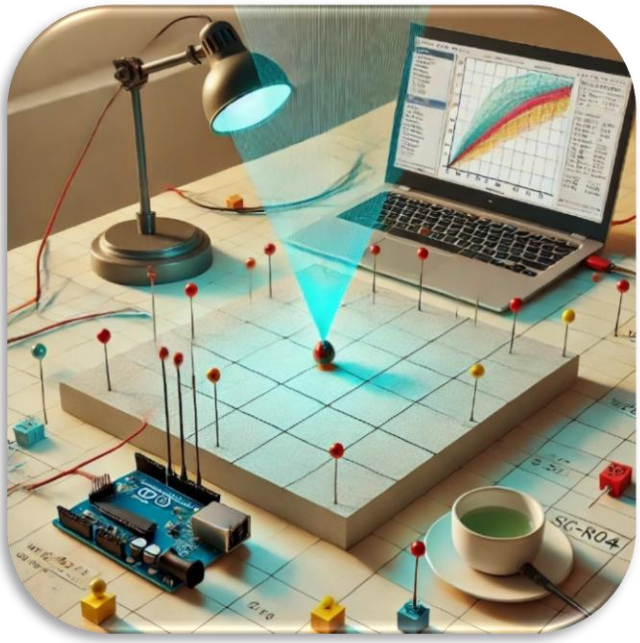
UNIVERSIDAD NACIONAL DEL ALTIPLANO

FACULTAD DE INGENIERÍA MECÁNICA ELÉCTRICA, ELECTRÓNICA Y SISTEMAS

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS



MEDICIÓN DE LA VELOCIDAD Y DIRECCIÓN EN UN PLANO USANDO VECTORES

DOCENTE: CARLOS CARCAUSTO QUISPE

ESTUDIANTES:

MARK GREGORY ARISACA TORRES

SEMESTRE: IIA

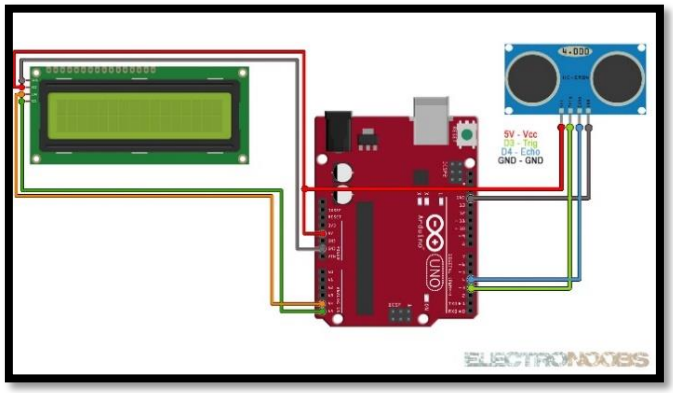
CODIGO: 241084

NUMERO DE ORDEN: 03

2024

1. **Conexión del Sensor HC-SR04 al Arduino:**

   - ✓ VCC (sensor) → 5V (Arduino)
   - ✓ Trig (sensor) → D3 (Arduino)
   - ✓ Echo (sensor) → D4 (Arduino)
   - ✓ GND (sensor) → GND (Arduino)



2. **Proceso de medición:** El sensor ultrasónico HC-SR04 mide la distancia de un objeto en movimiento en intervalos de tiempo fijos (1 segundo) en dos direcciones: primero en la dirección x y luego en la dirección y.
3. **Programa en Arduino:** El Arduino recoge las mediciones de distancia enviadas por el sensor ultrasónico y las transmite al puerto serial para su procesamiento en Python.

cpp

```
#define TRIG_PIN 3
#define ECHO_PIN 4

void setup() {
  Serial.begin(9600);
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
}

void loop() {
  long duration, distance;

  // Generar el pulso en el pin Trig
  digitalWrite(TRIG_PIN, LOW);
```

```
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);

  // Leer la duración del pulso en el pin Echo
  duration = pulseIn(ECHO_PIN, HIGH);

  // Convertir el tiempo en distancia (en cm)
  distance = duration * 0.034 / 2;

  // Enviar la distancia al puerto serial
  Serial.println(distance);

  delay(1000);  // Esperar 1 segundo entre mediciones
}
```

4. **Programa en Python:** El programa en Python procesa las distancias medidas en las direcciones x e y, calcula las componentes de la velocidad, la velocidad total, y el ángulo de dirección.

python

```
import serial
import time
import math

# Conectar con el puerto serial del Arduino
arduino = serial.Serial('COM3', 9600)  # Cambia 'COM3' por el puerto
correspondiente
time.sleep(2)

def obtener_distancias():
    distancias = []
    for _ in range(5):  # Capturar 5 mediciones de distancia
        distancia = float(arduino.readline().decode().strip())
        distancias.append(distancia)
        time.sleep(1)  # Intervalo de 1 segundo entre mediciones
    return distancias

# Función para calcular la velocidad
```

```
def calcular_velocidad(distancias_x, distancias_y, intervalo=1):
    velocidades_x = []
    velocidades_y = []

    for i in range(1, len(distancias_x)):
        vx = (distancias_x[i] - distancias_x[i - 1]) / intervalo
        vy = (distancias_y[i] - distancias_y[i - 1]) / intervalo
        velocidades_x.append(vx)
        velocidades_y.append(vy)

    return velocidades_x, velocidades_y

# Función para calcular la velocidad total y el ángulo
def calcular_vector(velocidades_x, velocidades_y):
    velocidades = []
    angulos = []

    for vx, vy in zip(velocidades_x, velocidades_y):
        velocidad = math.sqrt(vx**2 + vy**2)
        angulo = math.degrees(math.atan2(vy, vx))  # Ángulo en grados
        velocidades.append(velocidad)
        angulos.append(angulo)

    return velocidades, angulos

 # Programa principal
 if __name__ == "__main__":
    # Obtener distancias en las direcciones x e y
    print("Midiendo en la dirección X...")
    distancias_x = obtener_distancias()

    print("Midiendo en la dirección Y...")
    distancias_y = obtener_distancias()

    # Calcular velocidades en x e y
    velocidades_x, velocidades_y = calcular_velocidad(distancias_x,
distancias_y)

    # Calcular la velocidad total y el ángulo de movimiento
    velocidades, angulos = calcular_vector(velocidades_x,
velocidades_y)
```

```
print(f"Velocidades en X: {velocidades_x}")
print(f"Velocidades en Y: {velocidades_y}")
print(f"Velocidades resultantes: {velocidades}")
print(f"Ángulos de movimiento: {angulos}")
```