```
mport machine
import time
# Define the pin connected to the IR sensor
sensor_pin = machine.Pin(16, machine.Pin.IN)
# Initialize variables
pulse_count = 0
last_state = sensor_pin.value()
last_pulse_time = time.ticks_ms()
distance_per_pulse_cm = 0.2  # Distance between each hole in cm
# Variables for averaging
speed_samples = []  # Store recent speed samples for averaging
num_samples = 5       # Number of samples to average
# Function to handle interrupts
def sensor_interrupt(pin):
        global pulse_count, last_pulse_time
        pulse_count += 1
        last_pulse_time = time.ticks_ms()
# Attach interrupt handler to the sensor pin
sensor_pin.irq(trigger=machine.Pin.IRQ_FALLING, handler=sensor_interrupt)
try:
        while True:
        # Read the current state of the sensor
        current_state = sensor_pin.value()
        # Calculate elapsed time since last pulse
        current_time = time.ticks_ms()
        elapsed_time = time.ticks_diff(current_time, last_pulse_time)
        # Calculate speed when movement is detected
        if current_state != last_state:
        if elapsed_time > 0:
                speed_cm_per_sec = distance_per_pulse_cm / (elapsed_time / 1000.0)  #
Speed in cm/s
                speed_km_per_hr = speed_cm_per_sec * 3.6  # Convert speed to km/hr
                speed_samples.append(speed_km_per_hr)
                if len(speed_samples) > num_samples:
                speed_samples.pop(0)  # Remove oldest sample
          avg_speed = sum(speed_samples) / len(speed_samples)
                print("Speed (km/hr):", avg_speed)
        else:
                print("Speed (km/hr): 0")
        # Update last state
        last_state = current_state
        # If no movement detected
        if current_state == last_state and elapsed_time > 500:  # Adjust threshold as needed
        print("Speed (km/hr): 0")
```

```
        # Delay to prevent CPU hogging
        time.sleep(0.1)
except KeyboardInterrupt:
        # Clean up GPIO resources
        sensor_pin.irq(handler=None)
```