# Ultrasonic Sensor Timing Sequence

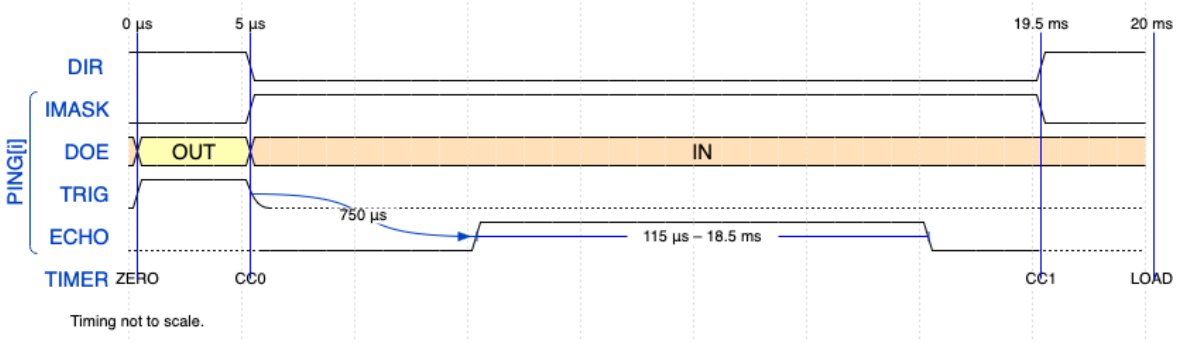


Figure 1: PING))) Ultrasonic Sensor Timing Sequence (Single-Sensor Cycle).

One full 20 ms measurement cycle for a single **Parallax PING))) Ultrasonic Distance Sensor** via an **SN74LVC245A** bidirectional level shifter. The MCU issues a 5 µs trigger, then reconfigures the shared I/O to input to measure the echo pulse width (time-of-flight).

| Signal | Name / Direction | Description |
|---|---|---|
| **DIR** | Buffer Direction | *SN74LVC245A* direction control.<br>1 = MCU → Sensor (trigger phase).<br>0 = Sensor → MCU (echo phase). |
| **PING**[$i$] | Sensor GPIO (Group) | Bidirectional MCU GPIO shared with ultrasonic sensor $i$.<br>Logical group encompassing IMASK, DOE, TRIG, and ECHO signals.<br>Manages the trigger, echo, and interrupt states for each sensor in the round-robin sequence. |
| **IMASK** | Interrupt Mask | Per-pin interrupt enable for the PING line.<br>1 = Interrupt enabled (ECHO edge capture active).<br>0 = Interrupt disabled (no capture events forwarded to NVIC). |
| **DOE** | Data Output Enable | MCU GPIO output-enable control for the PING line.<br>Input: idle and echo phases (interrupts disabled during idle).<br>Output: trigger phase. |
| **TRIG** | Trigger Pulse | 5 µs rising-edge pulse driven by MCU to initiate ultrasonic burst from the sensor. |
| **ECHO** | Echo Pulse | Return pulse width proportional to target distance.<br>Minimum: 115 µs<br>Maximum: 18.5 ms. |
| **TIMER** | Capture / Compare | Timer capture/compare reference points (ZERO, CC0, CC1, LOAD) aligned with echo pulse duration measurement. |

**Notes.**

1. Timer in edge-aligned up-count mode (LOAD = period limit).
2. Preload GPIO output before asserting DOE to avoid transients.
3. Round-robin across sensors with 20 ms spacing to avoid ultrasonic crosstalk.
4. SN74LVC245A provides 5 V ↔ 3.3 V logic compatibility.

# Multi-Sensor Schedule Cycle

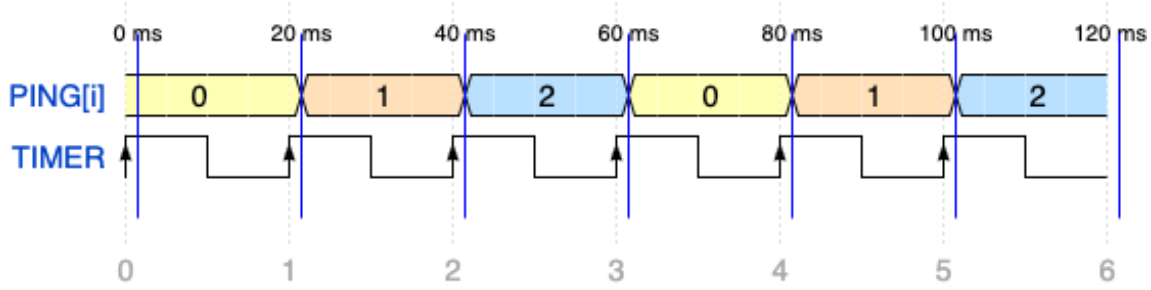


Figure 2: Round-robin schedule for three ultrasonic sensors.

The system operates three ultrasonic sensors in a round-robin sequence to prevent acoustic interference between transducers. Each sensor is allocated a dedicated 20 ms time slot encompassing its trigger, echo measurement, and a brief guard interval before the next sensor's cycle begins. This ensures that no two sensors emit ultrasonic bursts simultaneously.

| Slot | Time Window | Active Sensor |
|---|---|---|
| 0 | [ 0 ms, 20 ms) | PING[0] |
| 1 | [ 20 ms, 40 ms) | PING[1] |
| 2 | [ 40 ms, 60 ms) | PING[2] |

Cycle repeats every 60 ms, yielding an update rate of 16.7 Hz per sensor.

**Timing considerations.** As shown in the single-sensor timing diagram, each measurement cycle consists of a short trigger, a fixed holdoff interval, and an echo pulse lasting up to approximately 18.5 ms. Including these intervals yields a total cycle time of about 19.2 ms, so a 20 ms slot provides a small but sufficient margin to accommodate overhead and ensure all echoes have dissipated before the next sensor begins.

**Notes.**

1. Only one sensor is active per slot; the remaining sensors stay configured as inputs with their internal pull-downs enabled to prevent bus contention and maintain a defined low idle state.
2. The MCU cycles through sensors sequentially every 20 ms.
3. This schedule avoids cross-interference while maintaining a 50 Hz global timing base.
4. Additional sensors can be supported by proportionally extending the total cycle period.

# Differential-Drive Odometry

**Odometry** refers to estimating a robot's position $(x, y)$ and orientation $\theta$ over time by integrating wheel encoder measurements. In a differential-drive system, changes in left and right wheel displacements determine the robot's forward motion and rotation.

| Symbol | Name / Role | Definition / Notes |
|---|---|---|
| $N_L,\ N_R$ | Encoder counts (cum.) | Cumulative quadrature counts (ticks) for left/right wheels. |
| $\Delta N_L,\ \Delta N_R$ | Encoder count increments | Counts accumulated since the last update step (ticks). |
| $T$ | Ticks per revolution | Encoder ticks per wheel revolution (include quadrature mode and gearbox). |
| $R$ | Wheel radius | In meters. Used to convert ticks $\rightarrow$ distance. |
| $b$ | Wheel track | Lateral distance between wheel contact patches (m). |
| $(x,\ y,\ \theta)$ | Robot pose | World-frame position and heading; $\theta$ measured CCW from $+x$ (radians). |
| $\Delta s_L,\ \Delta s_R$ | Wheel travel (step) | Left/right wheel arc lengths over the last step: $\Delta s_L = \frac{2\pi R}{T}\,\Delta N_L, \quad \Delta s_R = \frac{2\pi R}{T}\,\Delta N_R$. |
| $\Delta s$ | Chassis forward travel (step) | Body-frame arc length over the last step: $\Delta s = \frac{1}{2}(\Delta s_R + \Delta s_L)$. |
| $\Delta\theta$ | Heading change (step) | Turn over the last step: $\Delta\theta = \frac{\Delta s_R - \Delta s_L}{b}$. |
| $s$ | Path length (cumulative) | Total forward distance traveled along the chassis arc. Update each step by $s \leftarrow s + \Delta s$. (Some notes use $s$ for a single step; here $s$ is cumulative and $\Delta s$ is per step.) |

1. **Wheel travel.** Converts encoder ticks into physical distance for each wheel. Each tick represents a small rotation, and multiplying by $\frac{2\pi R}{T}$ gives the linear displacement:

$$\Delta s_L = \frac{2\pi R}{T}\,\Delta N_L, \qquad \Delta s_R = \frac{2\pi R}{T}\,\Delta N_R$$

2. **Differential motion.** Computes how far the robot moved forward and how much it rotated. The average wheel travel gives forward motion, while the difference gives rotation about the centerline:

$$\Delta s = \frac{\Delta s_R + \Delta s_L}{2}, \qquad \Delta\theta = \frac{\Delta s_R - \Delta s_L}{b}$$

3. **Pose update (midpoint form).** Updates global position $(x, y)$ and heading $\theta$ using the arc traced during motion. The midpoint form accounts for the slight curvature of the path:

$$x \leftarrow x + \Delta s\cos\!\left(\theta + \tfrac{\Delta\theta}{2}\right), \quad y \leftarrow y + \Delta s\sin\!\left(\theta + \tfrac{\Delta\theta}{2}\right), \quad \theta \leftarrow \mathrm{wrap}(\theta + \Delta\theta)$$

   This effectively "walks" the pose forward by one small motion step.

4. **Exact arc formulation (optional).** When the robot turns significantly, you can compute the exact circular arc rather than the midpoint approximation. The instantaneous center of curvature (ICC) radius is:

$$R_{\mathrm{icc}} = \frac{\Delta s}{\Delta\theta} = \frac{b}{2}\,\frac{\Delta s_R + \Delta s_L}{\Delta s_R - \Delta s_L},$$

   and the pose update follows the true circular trajectory:

$$x \leftarrow x + R_{\mathrm{icc}}\,[\sin(\theta + \Delta\theta) - \sin\theta], \quad y \leftarrow y - R_{\mathrm{icc}}\,[\cos(\theta + \Delta\theta) - \cos\theta], \quad \theta \leftarrow \mathrm{wrap}(\theta + \Delta\theta)$$

   If $\Delta\theta \approx 0$ (straight motion), this simplifies to

$$x \leftarrow x + \Delta s\cos\theta, \qquad y \leftarrow y + \Delta s\sin\theta.$$

5. **Velocities (fixed timestep $\Delta t$).** Converts incremental motion into linear and angular velocities. These describe the robot's instantaneous motion:

$$v_L = \frac{2\pi R}{T}\frac{\Delta N_L}{\Delta t}, \qquad v_R = \frac{2\pi R}{T}\frac{\Delta N_R}{\Delta t}, \qquad v = \frac{v_R + v_L}{2}, \quad \omega = \frac{v_R - v_L}{b}$$

Integrating these gives continuous-time motion:

$$\dot{x} = v\cos\theta, \qquad \dot{y} = v\sin\theta, \qquad \dot{\theta} = \omega.$$

**Notes.**

1. Ensure encoder signs are consistent so both wheels forward give $\Delta\theta \approx 0$.
2. Include gearbox ratio and quadrature multiplication in $T$.
3. Measure $b$ between effective wheel contact points; small errors bias $\theta$.
4. Odometry drifts over time; fuse with IMU/vision for long runs.